

```
fi
exit 0
```

The most interesting script is *Xsession*. This version recognizes the special “failsafe” mode, specified in the translations in the *Xresources* file above, to provide an escape from the ordinary session:

```
#!/bin/sh

exec > $HOME/.xsession-errors 2>&1

case $# in
1)
    case $1 in
failsafe)
        exec xterm -geometry 80x24-0-0
        ;;
    esac
esac

startup=$HOME/.xsession
resources=$HOME/.Xresources

if [ -f $startup ]; then
    exec $startup
else
    if [ -f $resources ]; then
        xrdb -load $resources
    fi
    twm &
    exec xterm -geometry 80x24+10+10 -ls
fi
```

Controlling the Server

xdm controls local servers using POSIX signals. SIGHUP is expected to reset the server, closing all client connections and performing other clean up duties. SIGTERM is expected to terminate the server. If these signals do not perform the expected actions, *xdm* will not perform properly.

To control remote servers that are not using XDMCP, *xdm* searches the window hierarchy on the display and uses the protocol request `KillClient` in an attempt to clean up the terminal for the next session. This may not actually kill all of the clients, as only those which have created windows will be noticed. XDMCP provides a more certain mechanism; when *xdm* closes its initial connection, the session is over and the terminal is required to close all other connections.

Controlling xdm

xdm responds to two signals: SIGHUP and SIGTERM. When sent a SIGHUP, *xdm* rereads the configuration file, the access control file, and the servers file. For the servers file, it notices if entries have been added or removed. If a new entry has been added, *xdm* starts a session on the associated display. Entries that have been removed are disabled immediately, meaning that any session in progress will be terminated without notice, and no new session will be started.

When sent a SIGTERM, *xdm* terminates all sessions in progress and exits. This can be used when shutting down the system.

xdm attempts to mark the various subprocesses for *ps*(1) by editing the command-line argument list in place. Because *xdm* can't allocate additional space for this task, it is useful to start *xdm* with a reasonably long command line (using the full pathname should be enough). Each process that is servicing a display is marked *-display_name*.

Some Other Possibilities

You can also use *xdm* to run a single session at a time, using the 4.3 *init* options or other suitable daemon by specifying the server on the command line:

```
% xdm -server ":0 SUN-3/60CG4 local /usr/bin/X :0"
```

Or, you might have a file server and a collection of X terminals. The configuration for this could look identical to the sample above, except the *Xservers* file might look like:

```
london:0 VISUAL-19 foreign
paris:0 NCD-19 foreign
rome:0 NCR-TOWERVIEW3000 foreign
```

This would direct *xdm* to manage sessions on all three of these terminals. See the section "Controlling xdm" for a description of using signals to enable and disable these terminals in a manner reminiscent of *init*.

Limitations

xdm isn't very good at coexisting with other window systems. To use multiple window systems on the same hardware, you'll probably be more interested in *xinit*.

Files

/usr/lib/X11/xdm/xdm-config

The default configuration file.

/usr/lib/X11/xdm/Xaccess

The default access file, listing authorized displays. (Available as of Release 5.)

/usr/lib/X11/xdm/Xservers

The default server file, listing non-XDMCP servers to manage.

\$(HOME)/.Xauthority

User authorization file where *xdm* stores keys for clients to read.

xdm (continued)**X Display Manager**

/usr/lib/X11/xdm/chooser

The default chooser. (Available as of Release 5.)

/usr/bin/X11/xrdb

The default resource database loader.

/usr/bin/X11/X

The default server.

/usr/bin/X11/xterm

The default session program and failsafe client.

/usr/lib/X11/xdm/A<host>-<suffix>

The default place for authorization files.

See Also

X, xauth, xinit; Volume Eight, *X Window System Administrator's Guide*; the *Xsecurity* reference page in the MIT source distribution.

Author

Keith Packard, MIT X Consortium.

Name

`xdpr` – dump an X window directly to the printer.

Syntax

`xdpr` [*filename*] [*options*]

Description

`xdpr` runs the commands `xwd`, `xpr`, and `lpr(1)` or `lp(1)` to dump an X window, process it for a particular printer type, and print it out on the printer of your choice. This is the easiest way to get a printout of a window. By default, `xdpr` will print the largest possible representation of the window on the output page.

Options

The options for `xdpr` are the same as those for `xpr`, `xwd`, and `lpr(1)` or `lp(1)`. The most commonly-used options are described below; see the reference pages for these commands for detailed descriptions of the many other options available.

`-device printer_device`

Specifies the device on which the file is to be printed. Currently the following printers are supported:

`ln03` Digital LN03.

`la100` Digital LA100.

`ljet` HP LaserJet series and other monochrome PCL devices, such as ThinkJet, QuietJet, RuggedWriter, HP2560 series, and HP2930 series printers.

`pjet` HP PaintJet (color mode).

`pjetxl`

HP PaintJet XL Color Graphics Printer (color mode).

`pp` IBM PP3812.

`ps` PostScript printer.

As of Release 5, the default is `ps` (PostScript). (In prior releases, the default printer was the LN03.) `-device lw` (Apple LaserWriter) is equivalent to `-device ps` and is provided only for backwards compatibility.

`-display [host]:server[.screen]`

Specifies the name of the display to use. *host* is the hostname of the physical display, *server* specifies the server number, and *screen* specifies the screen number. Either or both of the *host* and *screen* elements to the display specification can be omitted. For example:

```
% xdpr -display your_node:0.1
```

prints a dump of an X window from screen 1 of server 0 on the display named by *your_node*. If the host is omitted, the local display is assumed. If the screen is omitted, screen 0 is assumed; the server and colon (:) are necessary in all cases.

-help Displays the list of options known to *xdpr*.

-Pprinter

Specifies a printer to send the output to. If a printer name is not specified here, *xdpr* (really, *lpr(1)* or *lp(1)*) will send your output to the printer specified by the `PRINTER` environment variable. Be sure that the type of the printer matches the type specified with the `-device` option.

xdpr also accepts the following argument:

filename

Specifies an existing file containing a window dump (created by *xwd*) to be printed instead of selecting an X window.

Any other arguments will be passed to the *xwd*, *xpr*, and *lpr(1)* or *lp(1)* commands as appropriate for each.

Environment Variables

`PRINTER`

Specifies which printer to use by default.

See Also

X, *xwd*, *xpr*, *xwud*, *lpr(1)*, *lp(1)*.

Authors

Paul Boutin, MIT Project Athena;
Michael R. Gretzinger, MIT Project Athena;
Jim Gettys, MIT Project Athena.

Name

xdpyinfo – display information utility for X.

Syntax

```
xdpyinfo [option]
```

Description

xdpyinfo is a utility for displaying information about an X server. It is used to examine the capabilities of a server, the predefined values for various parameters used in communicating between clients and the server, and the different types of screens and visuals that are available. See Chapter 8, *Other Clients*, for more information.

Option

xdpyinfo accepts the following option:

```
-display [host]:server[.screen]
```

Specifies the display about which *xdpyinfo* should display information. *host* is the hostname of the physical display, *server* specifies the server number, and *screen* specifies the screen number. By default, *xdpyinfo* displays information about all screens on the display. For example,

```
% xdpyinfo -display your_node:0.0
```

displays information about all screens of server 0 of the display named by *your_node*. If the hostname is omitted, the local display is assumed. If the screen is omitted, screen 0 is assumed. The server and colon (:) are necessary in all cases.

Sample Output

The following example shows the output produced by *xdpyinfo* when connected to a display that supports an 8-plane screen and a 1-plane screen.

```
name of display:      :0.0
version number:      11.0
vendor string:       MIT X Consortium
vendor release number:  4
maximum request size: 16384 longwords (65536 bytes)
motion buffer size:  0
bitmap unit, bit order, padding:  32, MSBFirst, 32
image byte order:    MSBFirst
number of supported pixmap formats:  2
supported pixmap formats:
    depth 1, bits_per_pixel 1, scanline_pad 32
    depth 8, bits_per_pixel 8, scanline_pad 32
keycode range:      minimum 8, maximum 129
focus: PointerRoot
number of extensions:  4
    SHAPE
    MIT-SHM
```

```

Multi-Buffering
MIT-SUNDRY-NONSTANDARD
default screen number: 0
number of screens: 2

screen #0:
dimensions: 1152x900 pixels (325x254 millimeters)
resolution: 90x90 dots per inch
depths (2): 1, 8
root window id: 0x8006e
depth of root window: 8 planes
number of colormaps: minimum 1, maximum 1
default colormap: 0x8006b
default number of colormap cells: 256
preallocated pixels: black 1, white 0
options: backing-store YES, save-unders YES
current input event mask: 0xd0801d
KeyPressMask      ButtonPressMask      ButtonReleaseMask
EnterWindowMask   ExposureMask          SubstructureRedirectMask
PropertyChangeMask ColormapChangeMask

number of visuals: 6
default visual id: 0x80065
visual:
visual id: 0x80065
class: PseudoColor
depth: 8 planes
size of colormap: 256 entries
red, green, blue masks: 0x0, 0x0, 0x0
significant bits in color specification: 8 bits
visual:
visual id: 0x80066
class: DirectColor
depth: 8 planes
size of colormap: 8 entries
red, green, blue masks: 0x7, 0x38, 0xc0
significant bits in color specification: 8 bits
visual:
visual id: 0x80067
class: GrayScale
depth: 8 planes
size of colormap: 256 entries
red, green, blue masks: 0x0, 0x0, 0x0
significant bits in color specification: 8 bits
visual:
visual id: 0x80068
class: StaticGray
depth: 8 planes
size of colormap: 256 entries
red, green, blue masks: 0x0, 0x0, 0x0

```

```

    significant bits in color specification:    8 bits
visual:
    visual id:    0x80069
    class:    StaticColor
    depth:    8 planes
    size of colormap:    256 entries
    red, green, blue masks:    0x7, 0x38, 0xc0
    significant bits in color specification:    8 bits
visual:
    visual id:    0x8006a
    class:    TrueColor
    depth:    8 planes
    size of colormap:    8 entries
    red, green, blue masks:    0x7, 0x38, 0xc0
    significant bits in color specification:    8 bits
number of mono multibuffer types:    6
    visual id, max buffers, depth:    0x80065, 0, 8
    visual id, max buffers, depth:    0x80066, 0, 8
    visual id, max buffers, depth:    0x80067, 0, 8
    visual id, max buffers, depth:    0x80068, 0, 8
    visual id, max buffers, depth:    0x80069, 0, 8
    visual id, max buffers, depth:    0x8006a, 0, 8
number of stereo multibuffer types:    0

screen #1:
    dimensions:    1152x900 pixels (325x254 millimeters)
    resolution:    90x90 dots per inch
    depths (1):    1
    root window id:    0x80070
    depth of root window:    1 plane
    number of colormaps:    minimum 1, maximum 1
    default colormap:    0x8006c
    default number of colormap cells:    2
    preallocated pixels:    black 1, white 0
    options:    backing-store YES, save-unders YES
    current input event mask:    0xd0801d
        KeyPressMask        ButtonPressMask        ButtonReleaseMask
        EnterWindowMask        ExposureMask        SubstructureRedirectMask
        PropertyChangeMask    ColormapChangeMask
    number of visuals:    1
    default visual id:    0x80064
    visual:
        visual id:    0x80064
        class:    StaticGray
        depth:    1 plane
        size of colormap:    2 entries
        red, green, blue masks:    0x0, 0x0, 0x0
        significant bits in color specification:    1 bits
    number of mono multibuffer types:    1

```

xcpyinfo *(continued)*

Display Information Utility

```
visual id, max buffers, depth: 0x80064, 0, 1  
number of stereo multibuffer types: 0
```

See Also

X, xwininfo, xprop, xrdp; Chapter 8, *Other Clients*.

Author

Jim Fulton, MIT X Consortium.

Name

xedit – simple text editor for X.

Syntax

xedit [*options*] [*filename*]

Description

xedit provides a window consisting of the following four areas:

Commands Section

A set of commands that allows you to exit *xedit*, save the file, or load a new file into the edit window.

Message Window

Displays *xedit* messages. In addition, this window can be used as a scratch pad.

Filename Display

Displays the name of the file currently being edited, and whether this file is *Read-Write* or *Read Only*.

Edit Window

Displays the text of the file that you are editing or creating.

Chapter 8, *Other Clients*, describes how to use the *xedit* client.

Command Buttons

- Quit** Quits the current editing session. If any changes have not been saved, *xedit* displays a warning message, allowing the user to save the file.
- Save** If file backups are enabled (see “Resources”), *xedit* stores a copy of the original, unedited file in <prefix>*filename*<suffix>, then overwrites the *filename* with the contents of the edit window. The *filename* is retrieved from the Text widget directly to the right of the Load button.
- Load** Loads the file named in the Text widget immediately to the right of this button and displays it in the Edit Window. If the currently displayed file has been modified, a warning message will ask the user to save the changes or to press Load again.

Editing

The Athena Text widget is used for the three sections of this application that allow text input, namely the Message Window, the Edit Window, and the window to the right of the command buttons, in which a filename can be entered.

The characters typed will go to the Text widget that the pointer is currently over. If the pointer is not over a Text widget, then the keypresses will have no effect on the application. This is also true for the special key sequences that pop-up dialog widgets; so, for example, typing CTRL-s in the filename widget (next to the command buttons) will enable searching in that widget, not the Edit Window (edit widget).

Both the Message Window and the Edit Window will create a scrollbar if the text to display is too large to fit in that window. Horizontal scrolling is not allowed by default, but can be turned on through the Text widget's resources. See Appendix G, *Widget Resources*, for more information.

The following list summarizes the editing commands recognized by *xedit* (i.e., by the Text widget).

Control-a	Move to the beginning of the current line.
Control-b	Move backward one character.
Control-d	Delete the next character.
Control-e	Move to the end of the current line.
Control-f	Move forward one character.
Control-h or Backspace	Delete the previous character.
Control-j, Control-m, Return, or LineFeed	New line.
Control-k	Kill the rest of this line. (Does not kill the carriage return at the end of the line. To do so, use Control-k twice. However, be aware that the second kill overwrites the text line in the kill buffer.)
Control-l	Redraw the window. (Also scrolls text so that cursor is positioned in the middle of the window.)
Control-n	Move down to the next line.
Control-o	Divide this line into two lines at this point and move the cursor back up.
Control-p	Move up to the previous line.
Control-r	Search and replace backward.
Control-s	Search and replace forward.
Control-t	Transpose characters. (Swap the characters immediately before and after the cursor.)
Control-u	Perform next command four times. For example, the sequence Control-u, Control-n moves the cursor down four lines.
Control-v	Move down to the next screenful of text.
Control-w	Kill the selected text.

Text Editor for X

xedit (continued)

Control-y	Insert the last killed text. (If the last killed text is a carriage return—see Control-k above—a blank line is inserted.)
Control-z	Scroll up the text one line.
Meta-<	Move to the beginning of the file.
Meta->	Move to the end of the file.
Meta-[Move backward one paragraph.
Meta-]	Move forward one paragraph.
Meta-b	Move backward one word.
Meta-d	Delete the next word.
Meta-D	Kill the next word.
Meta-f	Move forward one word.
Meta-h, Meta-Backspace, or Meta-Delete	Delete the previous word.
Meta-H, Meta-Shift-Backspace, or Meta-Shift-Delete	Kill the previous word.
Meta-i	Insert a file. A dialog box will appear in which you can type the desired filename.
Meta-k	Kill to the end of the paragraph.
Meta-q	Join lines to form a paragraph.
Meta-v	Move up to the previous screenful of text.
Meta-y	Insert the last selected text here. This command is the equivalent of clicking the second pointer button. See Chapter 5, <i>The xterm Terminal Emulator</i> , for more information about text selections.
Meta-z	Scroll down the text one line.
Delete	Delete the previous character.

Note that a translation in the application defaults file overrides the translation for the Return key for the text window in which a filename can be entered (next to the command buttons); in this window only, instead of starting a new line, Return moves the editing cursor to the end of the current line.

The Text widget fully supports the X selection and cut buffer mechanisms (described in Chapter 5, *The xterm Terminal Emulator*). Thus, you can cut and paste text in any of the sections of the *xedit* window that allow text input. You can also cut and paste text between *xedit* and any

other application (such as *xterm*) that supports text selections. See Chapter 5 for instructions on cutting and pasting text.

Options

xedit accepts all of the standard X Toolkit command-line options, which are listed on the X reference page. In addition, *xedit* accepts the following argument:

filename

Specifies the file that is to be loaded during start up. This is the file that will be edited. If a file is not specified, *xedit* lets you load a file or create a new file after it has started up.

Widget Hierarchy

In order to specify resources, it is useful to know the hierarchy of the widgets which compose *xedit*. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

```
Xedit  xedit
      Paned  paned
            Paned  buttons
                  Command  quit
                  Command  save
                  Command  load
                  Text     filename
            Label  bc_label
            Text   messageWindow

            Label  labelWindow
            Text   editWindow
```

See Appendix G, *Widget Resources* for a list of resources that can be set for the Athena widgets. (Note that the Text widget recognizes actions that control cursor movement, editing, text selection, etc.)

Resources

The available application resources are:

`enableBackups` (class `EnableBackups`)

Specifies that when edits made to an existing file are saved, *xedit* is to copy the original version of that file to `<prefix>filename<suffix>` before it saves the changes. The default value for this resource is “off,” stating that no backups should be created.

`backupNamePrefix` (class `BackupNamePrefix`)

Specifies a string that is to be prepended to the backup filename. The default is that no string shall be prepended.

`backupNameSuffix` (class `BackupNameSuffix`)

Specifies a string that is to be appended to the backup filename. The default is to append the string “.BAK”.

Actions

Many standard keyboard editing facilities are supported by the event bindings. You can map actions to key and pointer button events using the translation mechanism described in Chapter 11, *Setting Resources*. For the Text widget, the following actions are supported:

Cursor Movement	Action
forward-character	delete-next-character
backward-character	delete-previous-character
forward-word	delete-next-word
backward-word	delete-previous-word
forward-paragraph	delete-selection
backward-paragraph	backward-paragraph
beginning-of-line	selection
end-of-line	insert-selection
next-line	select-word
previous-line	select-all
next-page	select-start
previous-page	select-adjust
beginning-of-file	select-end
end-of-file	extend-start
scroll-one-line-up	extend-adjust
scroll-one-line-down	extend-end
new Line	miscellaneous
newline-and-indent	redraw-display
newline-and-backup	insert-file
newline	insert-char
kill	display-caret
kill-word	focus-in
backward-kill-word	focus-out
kill-selection	search
kill-to-end-of-line	multiply
kill-paragraph	form-paragraph
kill-to-end-of-paragraph	transpose-characters
no movement	no-op

- A page corresponds to the size of the Text window. For example, if the Text window is 50 lines in length, scrolling forward one page is the same as scrolling forward 50 lines.
- The `delete` action deletes a text item. The `kill` action deletes a text item and puts the item in the kill buffer (X cut buffer 1).
- The `insert-selection` action retrieves the value of a specified X selection or cut buffer, with fall-back to alternative selections or cut buffers.

Cursor Movement Actions`forward-character()``backward-character()`

These actions move the insert point forward or backward one character in the buffer. If the insert point is at the end (or beginning) of a line, this action moves the insert point to the next (or previous) line.

`forward-word()``backward-word()`

These actions move the insert point to the next or previous word boundary. A word boundary is defined as a space, a tab, or a carriage return.

`forward-paragraph()``backward-paragraph()`

These actions move the insert point to the next or previous paragraph boundary. A paragraph boundary is defined as two carriage returns in a row with only spaces or tabs between them.

`beginning-of-line()``end-of-line()`

These actions move to the beginning or end of the current line. If the insert point is already at the end or beginning of the line, no action is taken.

`next-line()``previous-line()`

These actions move the insert point up or down one line. If the insert point is currently n characters from the beginning of the line then it will be n characters from the beginning of the next or previous line. If n is past the end of the line, the insert point is placed at the end of the line.

`next-page()``previous-page()`

These actions move the insert point up or down one page in the file. One page is defined as the current height of the text widget. These actions always place the insert point at the first character of the top line.

`beginning-of-file()``end-of-file()`

These actions place the insert point at the beginning or end of the current text buffer. The text widget is then scrolled the minimum amount necessary to make the new insert point location visible.

```
scroll-one-line-up()
scroll-one-line-down()
```

These actions scroll the current text field up or down by one line. They do not move the insert point. Other than the scrollbars, this is the only way that the insert point may be moved off the visible text area. The widget will be scrolled so that the insert point is back on the screen as soon as some other action is executed.

Delete Actions

```
delete-next-character()
delete-previous-character()
```

These actions remove the character immediately after or before the insert point. If a carriage return is removed, the next line is appended to the end of the current line.

```
delete-next-word()
delete-previous-word()
```

These actions remove all characters between the insert point location and the next word boundary. A word boundary is defined as a space, a tab, or a carriage return.

```
delete-selection()
```

This action removes all characters in the current selection. The selection can be set with the selection actions.

Selection Actions

```
select-word()
```

This action selects the word in which the insert point is currently located. If the insert point is between words, it will select the previous word.

```
select-all()
```

This action selects the entire text buffer.

```
select-start()
```

This action sets the insert point to the current pointer location, where a selection then begins. If many of these selection actions occur quickly in succession then the selection count mechanism will be invoked.

```
select-adjust()
```

This action allows a selection started with the `select-start` action to be modified, as described above.

```
select-end(name[, name, ...])
```

This action ends a text selection that began with the `select-start` action, and asserts ownership of the selection or selections specified. A *name* can be a selection (e.g., PRIMARY) or a cut buffer (e.g., CUT_BUFFER0). Note that case is important. If no *names* are specified, PRIMARY is asserted.

```
extend-start()
```

This action finds the nearest end of the current selection, and moves it to the current pointer location.

`extend-adjust()`

This action allows a selection started with an `extend-start` action to be modified.

`extend-end(name[, name, . . .])`

This action ends a text selection that began with the `extend-start` action, and asserts ownership of the selection or selections specified. A *name* can be a selection (e.g., PRIMARY) or a cut buffer (e.g., CUT_BUFFER0). Note that case is important. If no *name* is given, PRIMARY is asserted.

`insert-selection(name[, name, . . .])`

This action retrieves the value of the first (left-most) named selection that exists or the cut buffer that is not empty. This action then inserts it into the Text widget at the current insert point location. A *name* can be a selection (e.g., PRIMARY) or a cut buffer (e.g., CUT_BUFFER0). Note that case is important.

newline Actions

`newline-and-indent()`

This action inserts a newline into the text and adds spaces to that line to indent it to match the previous line. (Note: this action still has a few bugs.)

`newline-and-backup()`

This action inserts a newline into the text *after* the insert point.

`newline()`

This action inserts a newline into the text *before* the insert point.

Kill Actions

`kill-word()`

`backward-kill-word()`

These actions act exactly like the `delete-next-word` and `delete-previous-word` actions, but they store the word that was killed into the kill buffer (CUT_BUFFER_1).

`kill-selection()`

This action deletes the current selection and stores the deleted text into the kill buffer (CUT_BUFFER_1).

`kill-to-end-of-line()`

This action deletes the entire line to the right of the insert point, and stores the deleted text into the kill buffer (CUT_BUFFER_1).

`kill-paragraph()`

This action deletes the current paragraph. If the insert point is between paragraphs, it deletes the paragraph above the insert point, and stores the deleted text into the kill buffer (CUT_BUFFER_1).

`kill-to-end-of-paragraph()`

This action deletes everything between the current insert point and the next paragraph boundary, and puts the deleted text into the kill buffer (CUT_BUFFER_1).

Miscellaneous Actions`redraw-display()`

This action recomputes the location of all the text lines on the display, scrolls the text to vertically center the line containing the insert point on the screen, clears the entire screen, and then redisplay it.

`insert-file([filename])`

This action activates the insert file pop-up. The *filename* option specifies the default filename to put in the filename buffer of the pop-up. If no *filename* is specified the buffer is empty at startup.

`insert-char()`

This action may be attached only to a key event. It calls `XLookupString` to translate the event into a (rebindable) Latin-1 character (sequence) and inserts that sequence into the text at the insert point position.

`insert-string(string[,string,...])`

This action inserts each *string* into the text at the insert point location. Any *string* beginning with the characters "0x" and containing only valid hexadecimal digits in the remainder is interpreted as a hexadecimal constant and the corresponding single character is inserted instead.

`display-caret(state,when)`

This action allows the insert point to be turned on and off. The *state* argument specifies the desired state of the insert point. This value may be any of the string values accepted for Boolean resources (e.g., on, True, off, False, etc.). If no arguments are specified, the default value is True. The *when* argument specifies, for `EnterNotify` or `LeaveNotify` events, whether or not the focus field in the event is to be examined. If the second argument is not specified, or specified as something other than `always`, then if the action is bound to an `EnterNotify` or `LeaveNotify` event, the action will be taken only if the focus field is True. An augmented binding that might be useful is:

```
*Text.Translations: #override \
    <FocusIn>:      display-caret (on) \n\
    <FocusOut>:     display-caret (off)
```

`focus-in()``focus-out()`

These actions do not currently do anything.

`search(direction,[string])`

This action activates the search popup. The *direction* must be specified as either `forward` or `backward`. The string is optional and is used as an initial value for the "Search for:" string.

`multiply(value)`

The multiply action allows the user to multiply the effects of many of the text actions. Thus the following action sequence:

```
multiply(10) delete-next-word()
```

will delete 10 words. It does not matter whether these actions take place in one event or many events. Using the default translations the key sequence Control-u, Control-d will delete 4 characters. Multiply actions can be chained; thus,

```
multiply(5) multiply(5)
```

is the same as:

```
multiply(25)
```

If the string `reset` is passed to the multiply action, the effects of all previous multiplies are removed and a beep is sent to the display.

`form-paragraph()`

This action removes all the carriage returns from the current paragraph and reinserts them so that each line is as long as possible, while still fitting on the current screen. Lines are broken at word boundaries if at all possible. This action currently works only on Text widgets that use ASCII text.

`transpose-characters()`

This action will switch the positions of the character to the left of the insert point and the character to the right of the insert point. The insert point will then be advanced one character.

`no-op([action])`

The no-op action makes no change to the text widget, and is used mainly to override translations. This action takes one optional argument. If this argument is `RingBell` then a beep is sent to the display.

Files

`/usr/lib/X11/app-defaults/Xedit`—Specifies required resources.

Restrictions

There is no undo function.

See Also

X, `xrdb`; Chapter 8, *Other Clients*; Appendix G, *Widget Resources*.

Copyright

Copyright © 1988, Digital Equipment Corporation. Copyright © 1989, Massachusetts Institute of Technology.

Text Editor for X

xedit *(continued)*

Author

Chris D. Peterson, MIT X Consortium.



Name

`xev` – print contents of X events.

Syntax

`xev` [*options*]

Description

`xev` creates a window and then asks the X server to send it notices, called *events*, whenever anything happens to the window (such as being moved, resized, typed in, clicked in, etc.). It is useful for seeing what causes events to occur and to display the information that they contain.

`xev` can be found in the *demos* directory in the X source tree. We feel it is sufficiently useful that we continue to document it here. See Chapter 14, *Setup Clients*, for instructions on using `xev` to assist in mapping keys.

Options

`xev` accepts the following options:

`-bs` *backing_store*

Specifies what kind of backing store to give the window (either `NotUseful`, `WhenMapped`, `Always`). The default is `NotUseful`.

`-bw` *pixels*

Specifies the width of the window border in pixels.

`-display` [*host*]:*server*[.*screen*]

Allows you to specify the host, server, and screen to connect to. *host* is the host-name of the physical display, *server* specifies the server number, and *screen* specifies the screen number. For example,

```
xev -display your_node:0.1
```

specifies screen 1 of server 0 on the display named by *your_node*. Either or both the *host* and *screen* elements can be omitted. If *host* is omitted, the local display is assumed. If *screen* is omitted, screen 0 is assumed (and the period is unnecessary). The colon and *server* are necessary in all cases.

`-geometry` *geometry*

The `xev` window is created with the specified size and location determined by the supplied geometry specification. The `-geometry` option can be (and often is) abbreviated to `-g`, unless there is a conflicting option that begins with “g”. The argument to the geometry option (*geometry*) is referred to as a “standard geometry string,” and has the form *widthxheight+*xoffset*+*yoffset**.

`-id` *window_ID*

Specifies that the window with the given ID should be monitored, instead of creating a new window.

Print X Events

xev (continued)

- name *string*
Specifies the name to assign to the created window.
- rv Specifies that the window should be in reverse video.
- s Specifies that save unders should be enabled on the window.

See Also

X, xmodmap, xwininfo, xdpinfo; Chapter 14, *Setup Clients*; Volume Zero, *X Protocol Reference Manual*; Volume One, *Xlib Programming Manual*.

Author

Jim Fulton, MIT X Consortium.

Name

`xfd` – Display all the characters in an X font.

Syntax

`xfd [options] -fn fontname`

Description

`xfd` creates a window containing the name of the font being displayed, a row of command buttons, several lines of text for displaying character metrics, and a grid containing one glyph per cell. The characters are shown in increasing order from left to right, top to bottom. The first character displayed at the top left will be character number 0 unless the `-start` option has been supplied, in which case the character with the number given in the `-start` option will be used.

The characters are displayed in a grid of boxes, each large enough to hold any single character in the font. Each character glyph is drawn using the `PolyText16` request (used by the Xlib routine `XDrawString16`). If the `-box` option is given, a rectangle will be drawn around each character, showing where an `ImageText16` request (used by the Xlib routine `XDrawImageString16`) would cause background color to be displayed.

The origin of each glyph is normally set so that the character is drawn in the upper-left corner of the grid cell. However, if a glyph has a negative left bearing or an unusually large ascent, descent, or right bearing (as is the case with the cursor font), some characters may not appear in their own grid cells. The `-center` option may be used to force all glyphs to be centered in their respective cells.

All the characters in the font may not fit in the window at once. To see the next page of glyphs, press the Next button at the top of the window. To see the previous page, press Prev. To exit `xfd`, press Quit.

Individual character metrics (index, width, bearings, ascent, and descent) can be displayed at the top of the window by pressing on the desired character.

The font name displayed at the top of the window is the full name of the font, as determined by the server. See *xlsfonts* for ways to generate lists of fonts, as well as more detailed summaries of their metrics and properties.

See Chapter 6, *Font Specification*, for instructions on using `xfd`.

Options

`xfd` accepts all of the standard X Toolkit command-line options, which are listed on the X reference page. In addition, `xfd` accepts the following application-specific options. (Note that the option `-fn font` is required.)

`-bc color`

Specifies the color to be used if `ImageText` boxes are drawn.

`-box` Displays a box outlining the area that would be filled with background color by an `ImageText` request.

Font Displayer**xfd** (continued)

- center
Indicates that each glyph should be centered in its grid square.
- fn *font*
Specifies the font to be displayed.
- start *char_num*
Specifies that character number *char_num* should be the first character displayed. (It appears in the upper left-hand corner of the grid.) This option is used to view characters at arbitrary locations in the font. The default is 0.

Resources

xfd accepts all of the Core resource names and classes as well as the following:

- boxChars (class BoxChars)
If True, displays a a box outlining the area that would be filled with background color by an ImageText request.
- boxColor (class Foreground)
Specifies that the given color is used for the ImageText boxes.
- cellColumns (class CellColumns)
Specifies number of columns in grid.
- cellRows (class CellRows)
Specifies number of rows in grid.
- centerChars (class CenterChars)
If True, each glyph is centered in its grid square.
- foreground (class Foreground)
Specifies the color to use for text and graphics within the window.
- metricsFormat (class MetricsFormat)
Defines the format of the text line displaying the metrics of a selected character.
- nocharFormat (class NocharFormat)
Defines the format of the text line displaying that no such character exists.
- rangeFormat (class RangeFormat)
Defines the format of the text line displaying the range of characters in the *xfd* window.
- startFormat (class StartFormat)
Defines the format of the text line displaying information about the starting character.
- selectFormat (class SelectFormat)
Defines the format of the text line displaying a selected character.

`startChar` (class `StartChar`)

Specifies that the given character number should be the first character displayed. (It appears in the upper left-hand corner of the grid.) This resource is used to view characters at arbitrary locations in the font. The default is 0.

Bugs

xfd should skip over pages full of non-existent characters.

See Also

X, `xfontsel`, `xlsfonts`, `xrdb`; Chapter 6, *Font Specification*.

Author

Jim Fulton, MIT X Consortium; previous program of the same name by Mark Lillibridge, MIT Project Athena.

Name

xfontsel – point-and-click interface for selecting display font names.

Syntax

```
xfontsel [options]
```

Description

xfontsel provides a simple way to display the fonts known to your X server, examine samples of each, and retrieve the X Logical Font Description (XLFD) full name for a font. (See Chapter 6, *Font Specification*, for instructions on using *xfontsel*.)

If `-pattern` is not specified, all fonts with XLFD 14-part names will be selectable. To work with only a subset of the fonts, specify `-pattern` followed by a partially or fully qualified font name. For example,

```
% xfontsel -pattern '*medium*' &
```

will select the subset of fonts that contain the string `medium` somewhere in their font name. Be careful about escaping wildcard characters in your shell.

If `-print` is specified on the command line, the selected font specifier will be written to standard output when the quit button is activated. Regardless of whether or not `-print` was specified, the font specifier may be made the PRIMARY text selection by activating the `select` button.

xfontsel handles scalable fonts as of Release 5. See `-noscaled` under “Options” later in this reference page and Chapter 6, *Font Specification*, for more information.

Clicking any pointer button in one of the XLFD field names will pop up a menu of the currently known possibilities for that field. If previous choices of other fields were made, only values for fonts which matched the previously selected fields will be selectable; to make other values selectable, you must deselect some other field(s) by choosing the “*” entry in that field. Unselectable values may be omitted from the menu entirely as a configuration option; see the `showUnselectable` resource, below. Whenever any change is made to a field value, *xfontsel* will assert ownership of the PRIMARY_FONT selection. Other applications (such as *xterm*) may then retrieve the selected font specification.

Scalable fonts come back from the server with zero for the pixel size, point size, and average width fields. Selecting a font name with a zero in these positions results in an implementation-dependent size. Any pixel or point size can be selected to scale the font to a particular size. Any average width can be selected to anamorphically scale the font (although you may find this challenging given the size of the average width menu).

Clicking the left pointer button in the `select` widget will cause the currently selected font name to become the PRIMARY text selection as well as the PRIMARY_FONT selection. Then you can paste the string into other applications. The `select` button remains highlighted to remind you of this fact, and dehighlights when some other application takes the PRIMARY selection away. The `select` widget is a toggle; pressing it when it is highlighted will cause *xfontsel* to release

the selection ownership and dehighlight the widget. Activating the `select` widget twice is the only way to cause *xfontsel* to release the PRIMARY_FONT selection.

Options

xfontsel accepts all of the standard X Toolkit command-line options, which are listed on the X reference page. In addition, *xfontsel* accepts the following application-specific options:

`-noscaled`

Disables the ability to select scaled fonts at arbitrary pixel or point sizes. This makes it clear which bitmap sizes are advertised by the server, and can avoid an accidental and sometimes prolonged wait for a font to be scaled. (Available as of Release 5.)

`-pattern fontname`

Specifies a subset of the available fonts, those with names that contain *fontname*, which can be a partial or full name.

`-print`

Specifies that the selected font will be written to standard output when the quit button is activated.

`-sample text`

Specifies the sample *text* to be used to display the selected font if the font is linearly indexed, overriding the default (the alphabetic characters; and the digits 0 through 9, if the character set includes them).

`-sample16 text`

Specifies the sample text to be used to display the selected font if the font is matrix encoded, overriding the default (see `-sample`). (Available as of Release 5.)

Resources

The application class is `XFontSel`. Most of the user interface is configured in the application defaults file; if this file is missing, a warning message will be printed to standard output and the resulting window will be nearly incomprehensible.

Most of the significant parts of the widget hierarchy are documented in the app-defaults file (normally `/usr/lib/X11/app-defaults/XFontSel`).

Application-specific resources:

`cursor` (class `Cursor`)

Specifies the cursor for the application window.

`pattern` (class `Pattern`)

Specifies the font name pattern for selecting a subset of available fonts. Equivalent to the `-pattern` option. Most useful patterns will contain at least one field delimiter, for example, `*-m-*` for monospaced fonts.

`pixelSizeList` (class `PixelSizeList`)

Specifies a list of pixel sizes to add to the pixel size menu, so that scalable fonts can be selected at those pixel sizes. The default list contains 7, 30, 40, 50, and 60. (Available as of Release 5.)

`pointSizeList` (class `PointSizeList`)

Specifies a list of point sizes (in units of tenths of points) to add to the point size menu, so that scalable fonts can be selected at those point sizes. The default list contains 250, 300, 350, and 400. (Available as of Release 5.)

`printOnQuit` (class `PrintOnQuit`)

If `True`, the currently selected font name is printed to standard output when the quit button is activated. Equivalent to the `-print` option.

`sampleText` (class `Text`)

Specifies the sample one-byte text to use for linearly indexed fonts. Each glyph index is a single byte, with a newline character separating lines. (Available as of Release 5.)

`sampleText16` (class `Text16`)

Specifies the sample two-byte text to use for matrix-encoded fonts. Each glyph index is two bytes, with a one-byte newline character separating lines. (Available as of Release 5.)

`scaledFonts` (class `ScaledFonts`)

The default value of `True` enables selection of arbitrary pixel and point sizes for scalable fonts. (Available as of Release 5.)

Widget-specific resources:

`showUnselectable` (class `ShowUnselectable`)

For each field menu, specifies whether or not to show values that are not currently selectable, based upon previous field selections. If shown, the unselectable values are clearly identified as such and do not highlight when the pointer is moved down the menu.

`fieldN.menu.options.showUnselectable` is the full instance name of this resource, while `MenuButton.SimpleMenu.Options.ShowUnselectable` is the full class name. In both cases, *N* is replaced with the field number (starting with the leftmost field numbered 0). The default is `True` for all but field 11 (average width of characters in font) and `False` for field 11. If you never want to see unselectable entries, `*menu.options.showUnselectable: False` is a reasonable thing to specify in a resource file.

Files

/usr/lib/X11/app-defaults/XFontSel
Specifies default resources.

See Also

xfd, *xrdb*; Chapter 6, *Font Specification*.

Bugs

Sufficiently ambiguous patterns can be misinterpreted and can lead to an initial selection string which may not correspond to what the user intended and which may cause the initial sample text output to fail to match the proffered string. Selecting any new field value will correct the sample output, though possibly resulting in no matching font.

The average width menu may be too long to be useful. (It may extend beyond the bounds of the screen.)

xfontsel should be able to return a font for the PRIMARY selection, not just a string.

Any change in a field value will cause *xfontsel* to assert ownership of the PRIMARY_FONT selection. Perhaps this should be parameterized.

When running on a slow machine, it is possible for the user to request a field menu before the font names have been completely parsed. An error message indicating a missing menu is printed to standard error, but otherwise nothing happens.

Author

Ralph R. Swick, Digital Equipment Corporation/MIT Project Athena.

Name

xhost – server access control program for X.

Syntax

xhost [*options*]

Description

The *xhost* program is used to add and delete hostnames (or user names) to and from the list allowed to make connections to the X server. In the case of hosts, this provides a rudimentary form of privacy control and security. It is only sufficient for a workstation (single user) environment, although it does limit the worst abuses. Environments that require more sophisticated measures should implement the user-based mechanism, or use the hooks in the protocol for passing authentication data to the server. (See Appendix A, *Managing Your Environment*, for an introduction to server access control and Volume Eight, *X Window System Administrator's Guide*, for a more complete discussion.)

The server initially allows network connections only from programs running on the same machine or from machines listed in the file */etc/Xn.hosts* (where *n* is the display number of the server). The *xhost* program is usually run either from a startup file or interactively to give access to other users.

Hostnames that are followed by two colons (::) are used in checking DECnet connections; all other hostnames are used for TCP/IP connections.

User names contain an at sign (@). When Secure RPC is being used, the network independent netname (e.g., “unix.uid@domainname”) can be specified, or a local user can be specified with just the username and a trailing at sign (e.g., “joe@”).

If no command-line options are given, a message indicating whether or not access control is currently enabled is printed on the standard output followed by the list of those allowed to connect. This is the only option that can be used from machines other than the controlling host.

Options

xhost accepts the command-line options described below. For security, the options that effect access control may only be run from the “controlling host.” For workstations, this is the same machine as the server. For X terminals, it is the login host.

[+]*name*

The given *name* (the plus sign is optional) is added to the list allowed to connect to the X server. The *name* can be a hostname or a user name.

-*name* The given *name* is removed from the list allowed to connect to the server. The *name* can be a hostname or a user name. Existing connections are not broken, but new connection attempts will be denied. Note that the current machine is allowed to be removed; however, further connections (including attempts to add it back) will not be permitted. Resetting the server (thereby breaking all connections) is the only way to allow local connections again.

- + Access is granted to everyone, even if they aren't on the list (i.e., access control is turned off).
- Access is restricted to only those on the list (i.e., access control is turned on).

Diagnostics

For each name added to the access control list, a line of the form “*name* being added to access control list” is printed. For each name removed from the access control list, a line of the form “*name* being removed from access control list” is printed.

Files

/etc/Xn.hosts

Bugs

You can't specify a display on the command line because `-display` indicates that you want to remove the machine named *display* from the access list.

This is not really a bug, but the X server stores network addresses, not hostnames. If somehow you change a host's network address while the server is still running, *xhost* must be used to add the new address and/or remove the old address.

See Also

X, Xserver, xauth, xdm; the section “Using `-display`” in Chapter 3, *Working in the X Environment*; Appendix A, *Managing Your Environment*; Volume Eight, *X Window System Administrator's Guide*.

Authors

Bob Scheifler, MIT Laboratory for Computer Science;
Jim Gettys, MIT Project Athena (DEC).

Name

xinit – X Window System initializer.

Syntax

```
xinit [[client] options] [-- [server_program] [display] options]
```

Description

The *xinit* program is used to start the X Window System server program and a first client program (usually a terminal emulator) on systems that cannot start X directly from */etc/init* or in environments that use multiple window systems. When this first client exits, *xinit* will kill the X server program and then terminate.

If no specific client program is given on the command line, *xinit* will look in the user's home directory for a file called *.xinitrc* to run as a shell script to start up other client programs. If no such file exists, *xinit* will use the following *xterm* command line as a default:

```
xterm -geometry +1+1 -n login -display :0
```

If no specific server program is given on the command line, *xinit* will look in the user's home directory for a file called *.xserverrc* to run as a shell script to start up the server. If no such file exists, *xinit* will use the following as a default server specification:

```
X :0
```

Note that this assumes that there is a server program called *X* in the current search path. However, servers are usually named *Xdisplaytype*, where *displaytype* is the type of graphics display that is driven by the server (for example, *Xsun*). The site administrator should therefore make a link to the appropriate type of server on the machine (see Chapter 2, *Getting Started*, and Volume Eight, *X Window System Administrator's Guide*), or create a shell script that runs *xinit* with the appropriate server.

Note that programs run by *.xinitrc* should be run in the background if they do not exit right away, so that they don't prevent other programs from starting up. However, the last long-lived program started (usually a window manager or terminal emulator) should be left in the foreground so that the script won't exit (which indicates that the user is done and that *xinit* should exit).

An alternate client and/or server may also be specified on the command line. The desired client program and its arguments should be given as the first command-line arguments to *xinit*. To specify a particular server program, append a double dash (--) to the *xinit* command line (after any client and arguments) followed by the desired server command.

Both the client program name and the server program name must begin with a slash (/) or a period (.); otherwise, they are treated as arguments to be appended to their respective startup lines. This makes it possible to add arguments (for example, foreground and background colors) without having to retype the whole command line.

If an explicit server name is not given and the first argument following the double dash (--) is a colon followed by a digit, *xinit* will use that number as the display number instead of zero. All remaining arguments are appended to the server command line.

Appendix A, *Managing Your Environment*, provides guidelines for writing a startup shell script. (See also “Examples” below.) See Volume Eight, *X Window System Administrator's Guide*, for instructions on running *xinit*.

Arguments

xinit accepts the following command-line arguments:

client

Specifies the client to be started with the server.

display

Specifies the number of the display on which to initialize the X Window System.

server_program

Specifies the server program to run.

Examples

xinit Will start up a server named *X* and run the user's *.xinitrc*, if it exists, or else start an *xterm*.

xinit -- /usr/bin/X11/Xqds :1

Is how one could start a specific type of server on an alternate display.

xinit -geometry 80x65+10+10 -fn 8x13 -j -fg white -bg navy

Will start up a server named *X* and will append the given arguments to the default *xterm* command. It will ignore *.xinitrc*.

xinit -e widgets -- ./Xsun -l -c

Will use the command *./Xsun -l -c* to start the server and will append the arguments *-e widgets* to the default *xterm* command.

xinit rsh fasthost cpupig -display workstation:1 -- :1 -a 2 -t 5

Will start a server named *X* on display 1 with the arguments *-a 2 -t 5*. It will then start a remote shell on the machine *fasthost* in which it will run the command *cpupig*, telling it to display back on the local workstation. Below is a sample *.xinitrc* that starts a clock, several terminals, and leaves an iconified *xterm* running as the “last” application. If the user kills this *xterm*, *X* shuts down.

```
#!/bin/sh
xrdb -load $HOME/.Xresources
xsetroot -solid gray &
mwm &
xclock -g 50x50-0+0 &
xterm -g 80x24+0+0 &
xterm -g 80x24+0+0 &
xterm -iconic
```

Sites that want to create a common startup environment could simply create a default `.xinitrc` that references a site-wide startup file:

```
#!/bin/sh
. /usr/local/lib/site.xinitrc
```

Another approach is to write a script that starts `xinit` with a specific shell script. Such scripts are usually named `x11`, `xstart`, or `startx` and are a convenient way to provide a simple interface for novice users:

```
#!/bin/sh
xinit /usr/local/bin/site.xinitrc -- /usr/bin/X11/Xhp :1
```

Environment Variables

XINITRC

Specifies an init file containing shell commands to start up the initial windows. By default, `.xinitrc` in the home directory will be used.

Files

- `.xinitrc` Script to start initial clients.
- `xterm` Client to run if `.xinitrc` does not exist.
- `.xserverrc`
Default script to start the server.
- `X` Server to run if `.xserverrc` does not exist.

See Also

X, Xserver, xterm, Appendix A, *Managing Your Environment*; Volume Eight, *X Window System Administrator's Guide*.

Author

Bob Scheifler, MIT Laboratory for Computer Science.

Name

`xkill` – kill a client by its X resource ID.

Syntax

`xkill` [*options*]

Description

`xkill` allows you to “kill” a client window or, more specifically, to force the X server to close the connection to the client. This program is very dangerous, but is useful for aborting programs that have displayed undesired windows on a user’s screen. If no window (resource) identification number is given with `-id`, `xkill` will display a special cursor as a prompt for the user to select a window to be killed. If a pointer button is pressed over a non-root window, the server will close its connection to the client that created the window and the window will be removed from the display. For a more detailed discussion of `xkill` and some problems inherent in “killing” a client window, see Chapter 8, *Other Clients*.

Options

`xkill` accepts the following application-specific options:

`-all` Indicates that all clients with top-level windows on the screen should be killed. `xkill` will ask you to select the root window with each of the currently defined buttons to give you several chances to abort. Use of this option is highly discouraged.

`-button` *number*

`-button` *any*

Specifies the number of the pointer button that should be used to select the window to kill. If the word *any* is specified, any button on the pointer can be used. By default, the first button in the pointer map (which is usually the leftmost button) is used.

`-display` [*host*]:*server*[.*screen*]

Allows you to specify the host, server, and screen to connect to. *host* is the host-name of the physical display, *server* specifies the server number, and *screen* specifies the screen number. For example,

```
xkill -display your_node:0.1
```

specifies screen 1 of server 0 on the display named by *your_node*. Either or both the *host* and *screen* elements to the display specification can be omitted. If *host* is omitted, the local display is assumed. If *screen* is omitted, screen 0 is assumed (and the period is unnecessary). The colon and *server* are necessary in all cases.

`-frame`

Indicates that `xkill` should ignore the standard conventions for finding top-level client windows (which are typically nested inside a window manager window), and simply believe that you want to kill direct children of the root. If you are running a window manager that provides titlebars or frames (such as *twm* or *mwm*), the children of the

Kill a Client**xkill** *(continued)*

root include these window decorations. Thus, when you select a client window to be killed, the window manager is killed instead.

-id resource

Allows you to specify the window to be killed using its resource (window) ID on the command line, rather than by selecting it with the pointer. If no window ID is specified, *xkill* will display a special cursor with which you should select a window to be killed.

Resources

xkill defines the following application resource:

Button

Specifies a pointer button number to use when selecting the window to be removed. If the word *any* is specified, any button on the pointer can be used.

See Also

X, *xwininfo*; Chapter 8, *Other Clients*; Volume One, *Xlib Programming Manual*.

Authors

Jim Fulton, MIT X Consortium;
Dana Chee, Bellcore.

xload

— Display Load Average —

Name

xload – display system load average.

Syntax

xload [*options*]

Description

The *xload* program displays a periodically updating histogram of the system load average. For instructions on using *xload*, see Chapter 3, *Working in the X Environment*, and Chapter 8, *Other Clients*.

Options

xload accepts all of the standard X Toolkit command-line options, which are listed on the X reference page. In addition, *xload* accepts the following application-specific options:

-hl *color*

-highlight *color*

Specifies the color of the scale lines.

-jumpscroll *pixels*

Specifies the number of pixels to shift the graph to the left when the graph reaches the right edge of the window. The default value is 1/2 the width of the current window. Smooth scrolling can be achieved by setting it to 1.

-label *string*

Specifies the text string for the label above the load average.

-lights

Causes *xload* to display the current load average by using the keyboard LEDs. For a load average of *n*, *xload* lights the first *n* keyboard LEDs. This option turns off the usual screen display. (Available as of Release 5.)

-nolabel

Specifies that no label be displayed above the load graph.

-scale *integer*

Specifies the minimum number of tick marks in the histogram, where one division represents one load average point. If the load goes above this number, *xload* will create more divisions, but it will never use fewer than this number. The default is 1.

-update *seconds*

Specifies the frequency in seconds at which *xload* updates its display. If the load average window is uncovered (by moving windows with a window manager or by the *xrefresh* program), the graph will also be updated. The minimum amount of time allowed between updates is 1 second. As of Release 5, the default is 10 seconds. (In Release 4, the default is 5 seconds.)

Display Load Average

xload (continued)

Resources

In addition to the resources available to each of the widgets used by *xload*, there is one resource defined by the application itself.

`showLabel` (class `Boolean`)

If `False`, then no label will be displayed.

You can set the following resource for the load widget:

`load.highlight` (class `Foreground`)

Specifies the color of the scale lines.

`load.jumpScroll` (class `JumpScroll`)

Specifies the number of pixels to shift the graph to the left when the graph reaches the right edge of the window. The default value is 1/2 the width of the current window. Smooth scrolling can be achieved by setting it to 1.

`load.minScale` (class `Scale`)

Specifies the minimum number of ticks that will be displayed. The default is 1.

`load.update` (class `Interval`)

Specifies the frequency in seconds at which *xload* updates its display. If the load average window is uncovered (by moving windows with a window manager or by the *xrefresh* program), the graph will also be updated. The minimum amount of time allowed between updates is 1 second. As of Release 5, the default is 10 seconds. (In Release 4, the default is 5 seconds.)

You can set the following resources for the label widget:

`*label.label` (class `String`)

Specifies that the given text string be used as a label above the load average. By default, the hostname is used.

Widget Hierarchy

In order to specify resources, it is useful to know the hierarchy of the widgets that compose *xload*. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name:

```

XLoad  xload
      Paned  paned
            Label  label
            StripChart  load

```

See Appendix G, *Widget Resources* for a list of resources that can be set for the Athena widgets.

Files

`/usr/lib/X11/app-defaults/XLoad`

Specifies required resources.

See Also

X, xrdb, mem(4); Chapter 3, *Working in the X Environment*; Chapter 8, *Other Clients*.

Bugs

This program requires the ability to open and read */dev/kmem*. Sites that do not allow general access to this file should make *xload* belong to the same group as */dev/kmem* and turn on the *set group id* permission flag.

Reading */dev/kmem* is inherently nonportable. Therefore, the routine used to read it (*get_load.c*) must be ported to each new operating system.

Authors

K. Shane Hartman (MIT-LCS) and Stuart A. Malone (MIT-LCS);
Additional features added by Jim Gettys (MIT-Athena), Bob Scheifler (MIT-LCS), Tony Della Fera (MIT-Athena), and Chris Peterson (MIT-LCS).

Name

xlogo – X Window System logo.

Synopsis

xlogo [*options*]

Description

The *xlogo* program displays the X Window System logo. This program is nothing more than a wrapper around the *undocumented* Athena Logo widget.

Options

xlogo accepts all of the standard X Toolkit command-line options, which are listed on the X reference page. (We've included some of the more commonly used Toolkit options later in this section.) In addition, *xlogo* accepts the following application-specific option:

-shape

Specifies that the *xlogo* window should be shaped to the X logo (rather than being rectangular). (Available as of Release 5.)

The following Toolkit options are commonly used:

-bg *color*

Specifies the color to use for the background of the window. The default is white. A correct color for the background is something like maroon.

-bd *color*

Specifies the color to use for the border of the window. The default is black.

-bw *pixels*

Specifies the width in pixels of the border surrounding the window.

-display [*host*]:*server*[*screen*]

Allows you to specify the physical display, server, and screen on which to create the *xlogo* window. See "Options" on the X reference page for an example of usage.

-fg *color*

Specifies the color to use for displaying the logo. The default is black. A correct color for the foreground is something like silver, which you can approximate with a shade of grey.

-geometry *geometry*

The *xlogo* window is created with the specified size and location determined by the supplied geometry specification. The `-geometry` option can be (and often is) abbreviated to `-g`, unless there is a conflicting option that begins with "g". The argument to the geometry option (*geometry*) is referred to as a "standard geometry string," and has the form *widthxheight±xoff±yoff*.

-rv Indicates that reverse video should be simulated by swapping the foreground and background colors.

`-xrm resourcestring`

Specifies a resource string to be used. This is especially useful for setting resources that do not have separate command-line options.

Resources

This program uses the Logo widget in the Athena Widget Set. It understands all of the Core and Simple widget resource names and classes as well as:

`foreground` (class `Foreground`)

Specifies the color for the logo. The default depends on whether `reverseVideo` is specified. If `reverseVideo` is specified the default is `XtDefaultBackground` (commonly white); otherwise, the default is `XtDefaultForeground` (commonly black).

`shapeWindow` (class `ShapeWindow`)

Specifies that the window be shaped to the X logo (rather than being rectangular). The default is `False`. (Available as of Release 5.)

The following Core resources are commonly used:

`width` (class `Width`)

Specifies the width of the logo. The default is 100 pixels.

`height` (class `Height`)

Specifies the height of the logo. The default is 100 pixels.

Widget Hierarchy

In order to specify resources, it is useful to know the hierarchy of the widgets that compose *xlogo*. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name:

```
XLogo xlogo
  Logo xlogo
```

Files

`/usr/lib/X11/app-defaults/XLogo`

Specifies required resources.

See Also

X, `xrdb`.

Authors

Ollie Jones of Apollo Computer and Jim Fulton of the X Consortium wrote the logo graphics routine, based on a graphic design by Danny Chong and Ross Chapman of Apollo Computer.

— List Interned Atoms

Name

`xlsatoms` – list interned atoms defined on the server.

Syntax

`xlsatoms` [*options*]

Description

`xlsatoms` lists the interned atoms. By default, all atoms starting from 1 (the lowest atom value defined by the protocol) are listed until an unknown atom is found. If an explicit range is given, `xlsatoms` will try all atoms in the range, regardless of whether or not any are undefined.

Options

`xlsatoms` accepts the following options:

`-display` [*host*]:*server*[.*screen*]

Specifies the display, server, and screen to use. *host* is the hostname of the physical display, *server* specifies the server number, and *screen* specifies the screen number. For example:

```
% xlsatoms -display your_node:0.1
```

specifies screen 1 of server 0 on the display named by *your_node*. Either or both the *host* and *screen* elements can be omitted. If *host* is omitted, the local display is assumed. If *screen* is omitted, screen 0 is assumed (and the period is unnecessary). The colon and *server* are necessary in all cases.

`-format` *printf_string*

Specifies a *printf*-style string used to list each atom *<value, name>* pair, printed in that order (*value* is an *unsigned long* and *name* is a *char **). `xlsatoms` will supply a newline at the end of each line. The default is `%ld\t%s`.

`-name` *string*

Specifies the name of an atom to list. If the atom does not exist, a message will be printed on the standard error.

`-range` [*low*]-[*high*]

Specifies the range of atom values to check. If *low* is not given, a value of 1 assumed. If *high* is not given, `xlsatoms` will stop at the first undefined atom at or above *low*.

See Also

X, Xserver, xprop.

Author

Jim Fulton, MIT X Consortium.

Name

xlsclients – list client applications running on a display.

Syntax

```
xlsclients [options]
```

Description

xlsclients is a utility for listing information about the client applications running on a display. It may be used to generate scripts representing a snapshot of the user's current session. Note, however, that *xlsclients* will not list a window manager process. See Chapter 8, *Other Clients*, for more information.

Options

xlsclients accepts the following options:

- a Specifies that clients on all screens should be listed. By default, only those clients on the default screen are listed.
- display [host]:server[.screen]
Allows you to specify the display, server, and screen to connect to. *host* is the host-name of the physical display, *server* specifies the server number, and *screen* specifies the screen number. For example,

```
% xlsclients -display your_node:0.1
```

specifies screen 1 of server 0 on the display named by *your_node*. Either or both the *host* and *screen* can be omitted. If *host* is omitted, the local display is assumed. If *screen* is omitted, screen 0 is assumed (and the period is unnecessary). The colon and *server* are necessary in all cases.
- l Requests a long listing showing the window name, icon name, and class hints in addition to the machine name and command string in the default listing.
- m *max_cmd_length*
Specifies the maximum number of characters in a command to list. The default is 10000.

See Also

X, xprop, xwininfo; Chapter 8, *Other Clients*.

Author

Jim Fulton, MIT X Consortium.

— List Available Fonts

Name

xlsfonts – list available fonts.

Syntax

xlsfonts [*options*] [-fn *pattern*]

Description

xlsfonts lists the fonts that match the given *pattern*. The wildcard character “*” may be used to match any sequence of characters (including none), and “?” to match any single character. If no pattern is given, “*” is assumed.

If you use wildcard characters as or within a fontname specification, the name must be quoted to prevent the characters from being expanded by the shell. See Chapter 6, *Font Specification*, for instructions on using *xlsfonts*.

Options

xlsfonts accepts the following options:

- 1 Indicates that listings should use a single column. This is the same as -n 1.
- C Indicates that listings should use multiple columns. This is the same as -n 0.
- display [*host*]:*server*[.*screen*]
 Allows you to specify the display, server, and screen to connect to. *host* is the host-name of the physical display, *server* specifies the server number, and *screen* specifies the screen number. For example,
 % xlsfonts -display *your_node*:0.1
 specifies screen 1 of server 0 on the display named by *your_node*. Either or both the *host* and *screen* can be omitted. If *host* is omitted, the local display is assumed. If *screen* is omitted, screen 0 is assumed (and the period is unnecessary).
- fn *pattern*
 Indicates that only fonts matching the specified *pattern* be listed.
- l [l [l]]
 Indicates that medium, long, and very long listings, respectively, should be generated for each font.
- m Indicates that long listings should also print the minimum and maximum bounds of each font.
- n *columns*
 Specifies the number of columns to use in displaying the output. By default, it will attempt to fit as many columns of font names as possible into the number of characters specified by -w *width*.
- o Specifies that *xlsfonts* should do an OpenFont (and QueryFont, if appropriate) rather than a ListFonts. This is useful if ListFonts or ListFontsWithInfo

fails to list a known font (as is the case with some scaled font systems). (Available as of Release 5.)

-u Specifies that the output should be left unsorted. (Available as of Release 5.)

-w *width*

Specifies the width in characters that should be used in figuring out how many columns to print. The default is 79.

See Also

X, Xserver, xset, xfd, xfontsel; Chapter 6, *Font Specification*.

Bugs

Doing `xlsfonts -l` can tie up your server for a very long time. This is really a bug with single-threaded, non-preemptable servers, not with this program.

Author

Mark Lillibridge, MIT Project Athena;
Jim Fulton, MIT X Consortium;
Phil Karlton, SGI.

Name

xlswins – server window list displayer for X.

Syntax

```
xlswins [options] [window_id]
```

Description

As of Release 5, this program is no longer included in the standard MIT X distribution.

You can access virtually the same information by running the command:

```
% xwininfo -tree
```

and then selecting the root window. For more information, see the *xwininfo* reference page and Chapter 8, *Other Clients*.

The *xlswins* reference page is included merely for continuity.

xlswins lists the window tree. By default, the root window is used as the starting point, although another window may be specified using the *window_id* option.

Options

xlswins accepts the following options:

```
-display [host]:server[.screen]
```

Specifies the display, server, and screen to use. *host* is the hostname of the physical display, *server* specifies the display server number, and *screen* specifies the screen number. For example:

```
% xlswins -display your_node:0.1
```

Either or both of the *host* and *screen* elements to the display specification can be omitted. If *host* is omitted, the local display is assumed. If *screen* is omitted, screen 0 is assumed (and the period is unnecessary). The colon and (display) *server* are necessary in all cases.

```
-format radix
```

Specifies the radix to use when printing out window IDs. Allowable values are: hex, octal, and decimal. The default is hex.

```
-indent number
```

Specifies the number of spaces that should be indented for each level in the window tree. The default is 2.

```
-l
```

Indicates that a long listing should be generated for each window. This includes a number indicating the depth, the geometry relative to the parent, and the location relative to the root window.

```
window_id
```

Specifies that the starting point for the window tree listing is the window *window_id*.

xlswins *(continued)*

List Window Tree

See Also

X, Xserver, xwininfo, xprop.

Author

Jim Fulton, MIT X Consortium.

Name

xmag – magnify parts of the screen.

Syntax

xmag [*options*]

Description

This reference page documents the Release 5 version of *xmag*, which operates significantly differently from prior releases. For instructions on using the Release 5 version, see Chapter 7, *Graphics Utilities*.

The *xmag* program allows you to magnify portions of the screen. If no explicit region is specified, a square with the pointer in the upper-left corner is displayed indicating the area to be enlarged. The area can be dragged out to the desired size by pressing pointer Button2. Once a region has been selected, a window is popped up showing an enlarged version of the region in which each pixel in the source image is represented by a small square of the same color. Pressing Button1 in the enlargement window shows the position and RGB value of the pixel under the pointer until the button is released.

The *xmag* client features five command buttons, described in the next section.

Two of the command buttons enable you to select multiple areas to be enlarged. You can open multiple instances of the magnification window or replace the current enlargement with a new image, using the `new` or `replace` command buttons, respectively.

Note that you can copy and paste images between *xmag* and *bitmap*. (See Chapter 7, *Graphics Utilities*, for more information.)

Resizing *xmag* resizes the current magnification area. *xmag* preserves the colormap, visual, and window depth of the source.

To quit *xmag*, type `q`, `Q`, or `Control-C` in the enlargement window, or select the `close` command button.

Command Buttons

There are five command buttons across the top of the *xmag* window. `Close` deletes the particular magnification instance (window). `Replace` brings up the rubber band selector again to select another region for this magnification window. `New` brings up the rubber band selector to create a new magnification window. `Cut` puts the magnification image into the PRIMARY selection. `Paste` copies the PRIMARY selection buffer into *xmag*. (You can copy and paste images between *xmag* and *bitmap*. See Chapter 7 for instructions.)

Options

xmag accepts all of the standard X Toolkit command-line options, which are listed on the X reference page. (We've included some of the more commonly used Toolkit options later in this section.) In addition, *xmag* accepts the following application-specific options:

-mag *mag_factor*

Specifies a factor by which the source region should be enlarged. The default magnification is 5. This is used with the size of the source to compute the default enlargement window size. (Specifying a size with `-geometry` can distort the program's results. See `-geometry` below.)

-source *geometry*

Specifies the size and/or location of the source region on the screen. By default, a 64 × 64 square is provided for the user to select an area of the screen. The size of the source is used with the desired magnification to compute the default enlargement window size. (Specifying a size with `-geometry` can distort the program's results. See `-geometry` below.)

The following standard X Toolkit options are commonly used with *xmag*:

-bg *color_or_pixel_value*

Specifies the name of the color to be used as the background of the enlargement window. If the name begins with a percent sign (%), it is interpreted to be an absolute pixel value. This is useful when displaying large areas, since pixels that are the same color as the background do not need to be painted in the enlargement. The default is to use the `BlackPixel` of the screen.

-display [*host*]:*server*[.*screen*]

Allows you to specify the display, server, and screen to use for both reading the screen and displaying the enlarged version of the image. *host* is the hostname of the physical display, *server* specifies the server number, and *screen* specifies the screen number. For example:

```
% xmag -display your_node:0.1
```

specifies screen 1 of server 0 on the display named by *your_node*. Either or both the *host* and *screen* elements to the display specification can be omitted. If *host* is omitted, the local display is assumed. If *screen* is omitted, screen 0 is assumed (and the period is unnecessary). The colon and *server* are necessary in all cases.

-fn *fontname*

This option specifies the name of a font to use when displaying pixel values (used when button 1 is pressed in the enlargement window).

-geometry *geometry*

The enlargement window is created with the specified size and location determined by the supplied geometry specification. See the X reference page for a description of usage.

Note that using the `-geometry` option with *xmag* can affect how the program works. By default, the size of the *xmag* window is computed from the size of the source region and the desired magnification. Therefore, specifying a size with `-geometry`

Magnify Screen Portions

xmag (continued)

can distort the program's results. As a general rule, you should only specify the location of the *xmag* window, as in:

```
% xmag -geometry -0-0 &
```

which places the window in the lower-right corner of the screen.

Resources

The *xmag* program defines the following application resources:

`mag`(class `Mag`)

Specifies the enlargement factor. See the `-mag` option for more information.

`source` (class `Source`)

Specifies the size and/or location of the source region on the screen. Takes as an argument the standard geometry string. See the `-source` option for more information.

`title` Specifies a string that may be used by the window manager (e.g., in a titlebar) when displaying this application.

The following X Toolkit resources are commonly used with *xmag*:

`background` (class `Background`)

Specifies the color or pixel value to be used for the background of the enlargement window.

`font` (class `Font`)

Specifies the name of the font to be used for the command buttons, and when displaying pixel values when the user presses button 1 in the enlargement window.

`foreground` (class `Foreground`)

Specifies the color or pixel value to be used for the foreground text of the enlargement window.

`geometry` (class `Geometry`)

Specifies the size and/or location of the enlargement window.

Widget Hierarchy

xmag uses the X Toolkit and the Athena Widget Set. The magnified image is displayed in the `Scale` widget. In order to specify resources, it is useful to know the hierarchy of the widgets that compose *xmag*. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

```
Xmag xmag
  RootWindow root
  TopLevelShell xmag
    Paned panel
      Paned pane2
        Command close
        Command replace
        Command new
```

```
Command select
Command paste
Label xmag label
Paned pane2
Scale scale
OverrideShell pixShell
Label pixLabel
```

See Appendix G, *Widget Resources*, for a list of resources that can be set for the Athena widgets.

See Also

X, bitmap, xwd; Chapter 7, *Graphics Utilities*.

Author

Release 5 version written by Dave Sternlicht and Davor Matic, MIT X Consortium;
Previous version by Jim Fulton, MIT X Consortium.

Name

xman – display manual pages.

Syntax

xman [*options*]

Description

xman is a manual page browser. The default size of the initial *xman* window is small so that you can leave it running throughout your entire login session. In the initial window there are three options: Help will pop up a window with online help, Quit will exit, and Manual Page will pop up a window with a manual page browser in it. You may pop up more than one manual page browser window at a time from a single execution of *xman*.

As of Release 5, typing Control-s will pop up a search window prompting for a specific manual page to display. If you know the exact name you want, type it and either press Return or click on the Manual Page button in the search window. The specific page will be displayed in a browser window. If you are not sure of the name, you can type in a guess and click on the Apropos button, which displays a list of reference pages containing the string you've entered.

For further information on using *xman*, please see Chapter 8, *Other Clients*, and read the online help information. The rest of this reference page will discuss customization of *xman*.

Customization

xman allows customization of both the directories to be searched for manual pages, and the name that each directory will map to in the Sections menu. *xman* determines which directories it will search by reading the MANPATH environment variable. If no MANPATH is found then the directory */usr/man* is searched on POSIX systems. This environment is expected to be a colon-separated list of directories for *xman* to search.

```
setenv MANPATH /mit/kit/man:/usr/man
```

By default, *xman* will search each of the following directories (in each of the directories specified in the user's MANPATH) for manual pages. If manual pages exist in that directory, then they are added to the list of manual pages for the corresponding menu item. A menu item is only displayed only for those sections that actually contain manual pages.

Directory	Section Name
man1	(1) User Commands
man2	(2) System Calls
man3	(3) Subroutines
man4	(4) Devices
man5	(5) File Formats
man6	(6) Games
man7	(7) Miscellaneous
man8	(8) Sys. Administration

Directory	Section Name
man1	(l) Local
mann	(n) New
mano	(o) Old

For instance, a user has three directories in her manual path and each contains a directory called *man3*. All these manual pages will appear, alphabetically sorted, when the user selects the menu item called (3) Subroutines. If there is no directory called *mano* in any of the directories in her MANPATH, or there are no manual pages in any of the directories called *mano*, then no menu item will be displayed for the section called (o) Old.

By using the *mandesc* file, a user or system manager is able to more closely control which manual pages will appear in each of the sections represented by menu items in the Sections menu. This functionality is available only on a section-by-section basis, and individual manual pages may not be handled in this manner (although generous use of symbolic links, *ln(1)*, will allow almost any configuration you can imagine).

The format of the *mandesc* file is a character followed by a label. The character determines which of the sections will be added under this label. For instance, suppose that you would like to create an extra menu item that contains all programmer subroutines. This label should contain all manual pages in both sections two and three. The *mandesc* file would look like this:

```
2Programmer Subroutines
3Programmer Subroutines
```

This will add a menu item to the Sections menu that would bring up a listing of all manual pages in sections two and three of the *UNIX Programmer's Manual*. Since the label names are *exactly* the same, they will be added to the same section. Note, however, that the original sections still exist.

If you want to completely ignore the default sections in a manual directory, then add the line:

```
no default sections
```

anywhere in your *mandesc* file. This keeps *xman* from searching the default manual sections *in that directory only*. As an example, suppose you want to do the same thing as above, but you don't think that it is useful to have the System Calls or Subroutines sections any longer. You would need to duplicate the default entries, as well as adding your new one.

```
no default sections
1(1) User Commands
2Programmer Subroutines
3Programmer Subroutines
4(4) Devices
5(5) File Formats
6(6) Games
7(7) Miscellaneous
8(8) Sys. Administration
```

l(l) Local
 n(n) New
 o(o) Old

xman will read any section that is of the form *man*<*character*>, where <*character*> is an uppercase or lowercase letter (they are treated distinctly) or a numeral (0-9). Be warned, however, that *man*(1) and *catman*(8) will not search directories that are nonstandard.

Options

xman accepts all of the standard X Toolkit command-line options, which are listed on the X reference page. In addition, *xman* accepts the following application-specific options:

-bothshown

Allows both the manual page and manual directory to be on the screen at the same time.

-geometry *geometry*

Sets the size and location of the top menu with the three buttons in it. This menu is created with the specified size and location determined by the supplied geometry specification. The **-geometry** option can be (and often is) abbreviated to **-g**, unless there is a conflicting option that begins with "g." The argument to the geometry option (*geometry*) is referred to as a "standard geometry string," and has the form *widthxheight±xoff±yoff*.

Note that *xman* allows you to use an equal sign followed by the *geometry* string as an alternative to **-geometry**.

-help Lists the valid options.

-helpfile *filename*

Specifies a helpfile to use other than the default.

-notopbox

Starts without the top menu with the three buttons in it.

-pagesize *geometry*

Sets the size and location of all the manual pages. See **-geometry** for the syntax of the *geometry* argument.

Resources

The *xman* program uses the following X Toolkit resources: foreground, background, width, height, borderWidth, and borderColor.

In addition, *xman* has application-specific resources that allow unique *xman* customizations.

bothShown (class Boolean)

Either True or False, specifies whether or not you want both the directory and the manual page shown at startup. The default is False.

- `directoryFontNormal` (class `Font`)
The font to use for the directory text.
- `directoryHeight` (class `DirectoryHeight`)
The height in pixels of the directory, when the directory and the manual page are shown simultaneously.
- `helpCursor` (class `Cursor`)
The cursor to use in the help window.
- `helpFile` (class `File`)
Use this rather than the system default help file.
- `manpageCursor` (class `Cursor`)
The cursor to use in the manual page window.
- `manualFontBold` (class `Font`)
The font to use for bold text in the manual pages.
- `manualFontItalic` (class `Font`)
The font to use for italic text in the manual pages.
- `manualFontNormal` (class `Font`)
The font to use for normal text in the manual pages.
- `pointerColor` (class `Foreground`)
The color of all the cursors (pointers) listed individually (`helpCursor`, `manpageCursor`, `topCursor`). The resource name was chosen to be compatible with *xterm*.
- `searchEntryCursor` (class `Cursor`)
The cursor to use in the search entry text widget.
- `topBox` (class `Boolean`)
Either `True` or `False`, determines whether the top box (containing the Help, Quit, and Manual Page buttons) or a manual page is put on the screen at startup. The default is `True`.
- `topCursor` (class `Cursor`)
The cursor to use in the top box.
- `verticalList` (class `Boolean`)
Either `True` or `False`, determines whether the directory listing is vertically or horizontally organized. The default is horizontal (`False`).

Widget Hierarchy

In order to specify resources, it is useful to know the hierarchy of the widgets that compose *xman*. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

```
Xman xman      (This widget is never used)
  TopLevelShell topbox
    Form form
```

```

Label topLabel
Command helpButton
Command quitButton
Command manpageButton
TransientShell search
  DialogWidgetClass dialog
    Label label
    Text value
    Command manualPage
    Command apropos
    Command cancel
TransientShell pleaseStandBy
  Label label
TopLevelShell manualBrowser
  Paned Manpage_Vpane
    Paned horizPane
      MenuButton options
      MenuButton sections
      Label manualBrowser
  Viewport directory
    List directory
    List directory
    .
    . (one for each section,
    . created "on the fly")
    .
  ScrollByLine manualPage
SimpleMenu optionMenu
  SmeBSB displayDirectory
  SmeBSB displayManualPage
  SmeBSB help
  SmeBSB search
  SmeBSB showBothScreens
  SmeBSB removeThisManpage
  SmeBSB openNewManpage
  SmeBSB showVersion
  SmeBSB quit
SimpleMenu sectionMenu
  SmeBSB <name of section>
  .
  . (one for each section)
  .
TransientShell search
  DialogWidgetClass dialog
    Label label
    Text value
    Command manualPage
    Command apropos

```

```

                                Command cancel
TransientShell pleaseStandBy
                                Label label
TransientShell likeToSave
                                Dialog dialog
                                Label label
                                Text value
                                Command yes
                                Command no
TopLevelShell help
Paned Manpage_Vpane
Paned horizPane
                                MenuButton options
                                MenuButton sections
                                Label manualBrowser
ScrollByLine manualPage
SimpleMenu optionMenu
SmeBSB displayDirectory
SmeBSB displayManualPage
SmeBSB help
SmeBSB search
SmeBSB showBothScreens
SmeBSB removeThisManpage
SmeBSB openNewManpage
SmeBSB showVersion
SmeBSB quit

```

See Appendix G, *Widget Resources*, for a list of resources that can be set for the Athena widgets.

Global Actions

xman defines all user interaction through global actions. This allows the user to modify the translation table of any widget and bind any event to the new user action. The list of actions supported by *xman* are:

`CreateNewManpage()`

Can be used anywhere; creates a new manual page display window.

`GotoPage(page)`

When used in a manual page display window, this action allows the user to move between a directory and manual page display. The *page* argument can be either `Directory` or `ManualPage`.

`PopupHelp()`

Can be used anywhere; pops up the help widget.

PopupSearch()

Can be used anywhere, except in a help window. It will cause the search popup to become active and visible on the screen, allowing the user to search for a manual page.

Quit()

Can be used anywhere; exits *xman*.

RemoveThisManpage()

Can be used in any manual page or help display window. When called, it will remove the window and clean up all resources associated with it.

SaveFormattedPage(action)

Can be used only in the `likeToSave` popup widget; tells *xman* whether to Save or Cancel a save of the manual page that has just been formatted.

Search(type, action)

Useful only when used in a search pop-up, this action will cause the search widget to perform the named search type on the string in the search popup's value widget. This action will also pop down the search widget. The *type* argument can be either `Apropos`, `Manpage`, or `Cancel`. If an *action* of `Open` is specified, then *xman* will open a new manual page to display the results of the search; otherwise, *xman* will attempt to display the results in the parent of the search popup.

ShowVersion()

May be called from any manual page or help display window, and will cause the informational display line to show the current version of *xman*.

Files

`<manpath directory>/man<character>`

`<manpath directory>/cat<character>`

`<manpath directory>/mandesc`

`/usr/lib/X11/app-defaults/Xman`

Specifies required resources.

`/tmp` *xman* creates temporary files in `/tmp` for all unformatted man pages and all apropos searches.

Environment Variables**MANPATH**

The search path for manual pages. Directories are separated by colons (e.g., `/usr/man:/mit/kit/man:/foo/bar/man`).

XAPPLRESDIR

A string that will have "Xman" appended to it. This string will be the full path name of a user application defaults file to be merged into the resource database after the system application defaults file, and before the resources that are attached to the display.

See Also

X, apropos(1), catman(8), ln(1), man(1); Chapter 8, *Other Clients*; Appendix G, *Widget Resources*.

Authors

Chris Peterson, MIT X Consortium, from the V10 version written by Barry Shein, formerly of Boston University.

Name

xmh – read and send mail with an X window interface to *mh*.

Syntax

xmh [-path *mailpath*] [-initial *foldername*] [-flag] [-toolkitoption]

Description

xmh provides a graphical user interface to the *mh* message handling system. To actually do things with your mail, *xmh* makes calls to the *mh* package. Electronic mail messages may be composed, sent, received, replied to, forwarded, sorted, and stored in folders.

Please don't be misled by the size of this document. It introduces many aspects of the Athena widget set and provides extensive mechanism for customization of the user interface. *xmh* really is easy to use.

For further information, see the Nutshell Handbook *MH & xmh: E-mail for Users & Programmers*, by Jerry Peek, also published by O'Reilly and Associates, Inc.

Options

xmh accepts all of the standard X Toolkit command-line options, which are listed on the X reference page. In addition, *xmh* accepts the following application-specific options:

-path *mailpath*

To specify an alternate collection of mail folders in which to process mail, use **-path** followed by the absolute pathname of the alternate mail directory. The default mail path is *\$HOME/Mail*; another mail path can be specified using the Path component in *\$HOME/.mh_profile*.

-initial *foldername*

Specifies an alternate folder that may receive new mail and is initially opened by *xmh*. The default initial folder is "inbox".

-flag Causes *xmh* to change the appearance of the appropriate folder buttons and to request the window manager to change the appearance of the *xmh* icon when new mail arrives. By default, *xmh* changes the appearance of the "inbox" folder button when new mail is waiting. You can use the application-specific resource `checkNewMail` to turn off this notification and the **-flag** option will still override it.

These three options have corresponding application-specific resources, named `MailPath`, `InitialFolder`, and `MailWaitingFlag`, which can be used in a resource file.

Installation

xmh requires that the user is already set up to use *mh*, version 6. First, see if there is a file called *.mh_profile* in your home directory. If it exists, check to see if it contains a line that starts with `Current-Folder`. If it does, you've been using version 4 or earlier of *mh*; to convert to version 6, you must remove that line. (Failure to do so causes spurious output to standard error, which, depending on your setup, can hang *xmh*.)

If you do not already have an *.mh_profile*, you can create one (and everything else you need) by typing `inc` to the shell. You should do this before using *xmh* to incorporate new mail.

For more information, refer to the *mh(1)* documentation and the Nutshell Handbook *MH & xmh*.

Much of the user interface of *xmh* is configured in the *Xmh* application defaults file; if this file was not installed properly a warning message will appear when *xmh* is used. The Release 5 version of *xmh* is backwards compatible with the R4 application defaults file.

The default value of the `SendBreakWidth` resource has changed since R4.

Basic Screen Layout

xmh starts out with a single window, divided into four main areas:

- Six buttons with pull-down command menus.
- A collection of buttons, one for each top level folder. New users of *mh* will have two folders, “drafts” and “inbox”.
- A listing, or Table of Contents, of the messages in the open folder. Initially, this will show the messages in “inbox”.
- A view of one of your messages. Initially this is blank.

xmh and the Athena Widget Set

xmh uses the X Toolkit Intrinsics and the Athena widget set. Many of the features described below (scrollbars, buttonboxes, etc.) are actually part of the Athena widget set and are described here only for completeness. For more information, see Appendix G, *Widget Resources*.

Scrollbars

Some parts of the main window will have a vertical area on the left containing a gray bar. This area is a *scrollbar*. Scrollbars are used whenever the data in a window takes up more space than can be displayed. The gray bar indicates what portion of your data is visible. If the entire length of the area is gray, then you are looking at all your data. If only the first half is gray, then you are looking at the top half of your data. The message viewing area will have a horizontal scrollbar if the text of the message is wider than the viewing area.

You can use the pointer in the scrollbar to change what part of the data is visible. If you click the second pointer button, then the top of the gray area will move to where the pointer is, and the corresponding portion of data will be displayed. If you hold down the second pointer button, you can drag around the gray area. This makes it easy to get to the top of the data: just press with the second button, drag off the top of the scrollbar, and release.

If you click with the first pointer button, then the data to the right of the pointer will scroll to the top of the window. If you click with the third pointer button, then the data at the top of the window will scroll down to where the pointer is.

Buttonboxes, Buttons, and Menus

Any area containing many words or short phrases, each enclosed in a rectangular or rounded boundary, is called a *buttonbox*. Each rectangle or rounded area is actually a button that you can press by moving the pointer onto it and pressing pointer button 1. If a given buttonbox has more buttons in it than can fit, it will be displayed with a scrollbar, so you can always scroll to the button you want.

Some buttons have pull-down menus. Pressing the pointer button while the pointer is over one of these buttons will pull down a menu. Continuing to hold the button down while moving the pointer over the menu, called dragging the pointer, will highlight each selectable item on the menu as the pointer passes over it. To select an item in the menu, release the pointer button while the item is highlighted.

Adjusting the Relative Sizes of Areas

If you're not satisfied with the sizes of the various areas of the main window, they can easily be changed. Near the right edge of the border between each region is a black box, called a *grip*. Simply point to that grip with the pointer, press a pointer button, drag up or down, and release. Exactly what happens depends on which pointer button you press:

If you drag with pointer button 2, then only that border will move. This mode is simplest to understand, but is the least useful.

If you drag with pointer button 1, then you are adjusting the size of the window above. *xmh* will attempt to compensate by adjusting some window below it.

If you drag with pointer button 3, then you are adjusting the size of the window below. *xmh* will attempt to compensate by adjusting some window above it.

All windows have a minimum and maximum size; you will never be allowed to move a border past the point where it would make a window have an invalid size.

Processing Your Mail

This section will define the concepts of the selected folder, current folder, selected message(s), current message, selected sequence, and current sequence. Each *xmh* command is introduced.

For use in customization, action procedures corresponding to each command are given; these action procedures can be used to customize the user interface, particularly the keyboard accelerators and the functionality of the buttons in the optional buttonbox created by the application resource `CommandButtonCount`.

Folders and Sequences

A folder contains a collection of mail messages, or is empty. *xmh* supports folders with one level of subfolders.

The selected folder is whichever folder name appears in the bar above the folder buttons. Note that this is not necessarily the same folder that is being viewed. To change the selected folder, just press on the desired folder button; if that folder has subfolders, select a folder from the pull-down menu.

The Table of Contents, or toc, lists the messages in the viewed folder. The titlebar above the Table of Contents displays the name of the viewed folder.

The toc titlebar also displays the name of the viewed sequence of messages within the viewed folder. Every folder has an “all” sequence, which contains all the messages in the folder, and initially the toc titlebar will show “inbox:all”.

Folder Commands

The Folder command menu contains commands of a global nature:

Open Folder

Displays the data in the selected folder. Thus, the selected folder also becomes the viewed folder. The action procedure corresponding to this command is `XmhOpenFolder([foldername])`. It takes an optional argument as the name of a folder to select and open; if no folder is specified, the selected folder is opened. It may be specified as part of an event translation from a folder menu button or from a folder menu, or as a binding of a keyboard accelerator to any widget other than the folder menu buttons or the folder menus.

Open Folder in New Window

Displays the selected folder in an additional main window. Note, however, that you may not reliably display the same folder in more than one window at a time, although *xmh* will not prevent you from trying. The corresponding action is `XmhOpenFolderInNewWindow()`.

Create Folder

Creates a new folder. You will be prompted for a name for the new folder; to enter the name, move the pointer to the blank box provided and type. Subfolders are created by specifying the parent folder, a slash, and the subfolder name. For example, to create a folder named “xmh” which is a subfolder of an existing folder named “clients”, type “clients/xmh”. Click on the Okay button when finished, or just press Return; click on Cancel to cancel this operation. The action corresponding to Create Folder is `XmhCreateFolder()`.

Delete Folder

Destroys the selected folder. You will be asked to confirm this action (see “Confirmation Windows”). Destroying a folder will also destroy any subfolders of that folder. The corresponding action is `XmhDeleteFolder()`.

Close Window

Exits *xmh*, after first confirming that you won’t lose any changes; or, if selected from any additional *xmh* window, simply closes that window. The corresponding action is `XmhClose()`.

Highlighted Messages, Selected Messages and the Current Message

It is possible to highlight a set of adjacent messages in the area of the Table of Contents. To highlight a message, click on it with pointer button 1. To highlight a range of messages, click on the first one with pointer button 1 and on the last one with pointer button 3; or press pointer

X Interface to mh**xmh** (continued)

button 1, drag, and release. To extend a range of selected messages, use pointer button 3. To highlight all messages in the table of contents, click rapidly three times with pointer button 1. To cancel any selection in the table of contents, click rapidly twice.

The selected messages are the same as the highlighted messages, if any. If no messages are highlighted, then the selected messages are considered the same as the current message.

The current message is indicated by a “+” next to the message number. It usually corresponds to the message currently being viewed. When a message is viewed, the titlebar above the view will identify the message.

Table of Contents Commands

The Table of Contents command menu contains commands that operate on the open, or viewed, folder.

Incorporate New Mail

Adds any new mail received to viewed folder and sets the current message to be the first new message. This command is selectable in the menu and will execute only if the viewed folder is allowed to receive new mail. By default, only “inbox” is allowed to receive new mail. The corresponding action is `XmhIncorporateNewMail()`.

Commit Changes

Executes all deletions, moves, and copies that have been marked in this folder. The corresponding action is `XmhCommitChanges()`.

Pack Folder

Renumbers the messages in this folder so they start with 1 and increment by 1. The corresponding action is `XmhPackFolder()`.

Sort Folder

Sorts the messages in this folder in chronological order. (As a side effect, this may also pack the folder.) The corresponding action is `XmhSortFolder()`.

Rescan Folder

Rebuilds the list of messages. This can be used whenever you suspect that *xmh*'s idea of what messages you have is wrong. (In particular, this is necessary if you change things using straight *mh* commands without using *xmh*.) The corresponding action is `XmhForceRescan()`.

Message Commands

The Message command menu contains commands that operate on the selected message(s), or if there are no selected messages, the current message.

Compose Message

Composes a new message. A new window will be brought up for composition; a description of it is given in the “Composition Windows” section below. This command does not affect the current message. The corresponding action is `XmhComposeMessage()`.

View Next Message

Views the first selected message. If no messages are highlighted, views the current message. If the current message is already being viewed, views the first unmarked message after the current message. The corresponding action is `XmhViewNextMessage()`.

View Previous

Views the last selected message. If no messages are highlighted, views the current message. If current message is already being viewed, views the first unmarked message before the current message. The corresponding action is `XmhViewPrevious()`.

Delete Marks the selected messages for deletion. If no messages are highlighted, this command marks the current message for deletion and automatically displays the next unmarked message. The corresponding action is `XmhMarkDeleted()`.

Move Marks the selected messages to be moved into the currently selected folder. (If the selected folder is the same as the viewed folder, this command will just beep.) If no messages are highlighted, this command marks the current message to be moved and displays the next unmarked message. The corresponding action is `XmhMarkMove()`.

Copy as Link

Marks the selected messages to be copied into the selected folder. (If the selected folder is the same as the viewed folder, this command will just beep.) If no messages are highlighted, marks the current message to be copied. Note that messages are actually linked, not copied; editing a message copied by *xmh* will affect all copies of the message. The corresponding action is `XmhMarkCopy()`.

Unmark Removes any of the above three marks from the selected messages, or the current message, if none is highlighted. The corresponding action is `XmhUnmark()`.

View in New

Creates a new window containing only a view of the first selected message, or the current message, if none is highlighted. The corresponding action is `XmhViewInNewWindow()`.

Reply Creates a composition window in reply to the first selected message, or the current message, if none is highlighted. The corresponding action is `XmhReply()`.

Forward Creates a composition window whose body is initialized to contain an encapsulation of the selected messages, or the current message if none is highlighted. The corresponding action is `XmhForward()`.

Use as Composition

Creates a composition window whose body is initialized to be the contents of the first selected message, or the current message if none is selected. Any changes you make in the composition will be saved in a new message in the "drafts" folder, and will not change the original message. However, there is an exception to this rule. If the

message to be used as composition was selected from the “drafts” folder (see “Bugs”), the changes will be reflected in the original message (see “Composition Windows”). The corresponding action is `XmhUseAsComposition()`.

Print Prints the selected messages, or the current message if none is selected. *xmh* normally prints by invoking the *enscript*(1) command, but this can be customized with the *xmh* resource `PrintCommand`. The corresponding action is `XmhPrint()`.

Sequence Commands

The Sequence command menu contains commands pertaining to message sequences (See “Message Sequences”), and a list of the message sequences defined for the currently viewed folder. The selected message sequence is indicated by a check mark in its entry in the margin of the menu. To change the selected message sequence, select a new message sequence from the sequence menu.

Pick Messages

Defines a new message sequence. The corresponding action is `XmhPickMessages()`.

The following menu entries will be sensitive only if the current folder has any message sequences other than the “all” message sequence.

Open Sequence

Changes the viewed sequence to be the same as the selected sequence. The corresponding action is `XmhOpenSequence()`.

Add to Sequence

Adds the selected messages to the selected sequence. The corresponding action is `XmhAddToSequence()`.

Remove from Sequence

Removes the selected messages from the selected sequence. The corresponding action is `XmhRemoveFromSequence()`.

Delete Sequence

Removes the selected sequence entirely. The messages themselves are not affected; they are simply no longer grouped together to define a message sequence. The corresponding action is `XmhDeleteSequence()`.

View Commands

Commands in the View menu and in the buttonboxes of view windows (which result from the Message command View in New) correspond in functionality to commands of the same name in the Message menu, but they operate on the viewed message rather than the selected messages or current message.

Close Window

When the viewed message is in a separate view window, this command will close the view, after confirming the status of any unsaved edits. The corresponding action procedure is `XmhCloseView()`.

- Reply** Creates a composition window in reply to the viewed message. The related action procedure is `XmhViewReply()`.
- Forward** Creates a composition window whose body is initialized to contain the contents of the viewed message. The corresponding action is `XmhViewForward()`.
- Use As Composition**
Creates a composition window whose body is initialized to be the contents of the viewed message. Any changes made in the composition window will be saved in a new message in the “drafts” folder and will not change the original message. An exception: if the viewed message was selected from the “drafts” folder (see “Bugs”), the original message is edited. The action procedure corresponding to this command is `XmhViewUseAsComposition()`.
- Edit Message**
Enables the direct editing of the viewed message. The action procedure is `Xmh-EditView()`.
- Save Message**
This command is insensitive until the message has been edited; when activated, edits will be saved to the original message in the view. The corresponding action is `Xmh-SaveView()`.
- Print** Prints the viewed message. *xmh* prints by invoking the `enscript(1)` command, but this can be customized with the application-specific resource `PrintCommand`. The corresponding action procedure is `XmhPrintView()`.
- Delete** Marks the viewed message for deletion. The corresponding action is `XmhView-MarkDelete()`. (Available as of Release 5.)

Options Menu

The Options menu contains one entry.

Read in Reverse

When selected, a check mark appears in the margin of this menu entry. Read in Reverse will switch the meaning of the next and previous messages and will increment to the current message marker in the opposite direction. This is useful if you want to read your messages in the order of most recent first. The option acts as a toggle; select it from the menu a second time to undo the effect. The check mark appears when the option is selected.

Composition Windows

Composition windows are created by selecting Compose Message from the Message menu, or by selecting Reply, Forward, or Use as Composition from either the Message or View menu. Aside from the normal text editing functions, there are six command buttons associated with composition windows:

X Interface to mh**xmh** (continued)**Close Window**

Closes this composition window. If changes have been made since the most recent Save or Send, you will be asked to confirm losing them. The corresponding action is `XmhCloseView()`.

Send Sends this composition. The corresponding action is `XmhSend()`.

New Headers

Replaces the current composition with an empty message. If changes have been made since the most recent Send or Save, you will be asked to confirm losing them. The corresponding action is `XmhResetCompose()`.

Compose Message

Brings up another new composition window. The corresponding action is `XmhComposeMessage()`.

Save Message

Saves this composition in your drafts folder. Then you can safely close the composition. At some future date, you can continue working on the composition by opening the drafts folder, selecting the message, and using the Use as Composition command. The corresponding action is `XmhSave()`.

Insert Inserts a related message into the composition. If the composition window was created with a Reply command, the related message is the message being replied to; otherwise, no related message is defined and this button is insensitive. The message may be filtered before being inserted; see `ReplyInsertFilter` under “Application-specific Resources” for more information. The corresponding action is `XmhInsert()`.

Accelerators

Accelerators are shortcuts. They allow you to invoke commands without using the menus, either from the keyboard or by using the pointer.

xmh defines pointer accelerators for common actions: to select and view a message with a single click, use pointer button 2 on the message’s entry in the table of contents; to select and open a folder or a sequence in a single action, make the folder or sequence selection with pointer button 2.

To mark the highlighted messages to be moved to a folder in a single action, or current message if none has been highlighted, use pointer button 3 to select the target folder and simultaneously mark the messages. Similarly, selecting a sequence with pointer button 3 will add the highlighted or current message(s) to that sequence. In both of these operations, the selected folder or sequence and the viewed folder or sequence are not changed.

xmh defines the following keyboard accelerators over the surface of the main window, except in the view area while editing a message:

Meta-l	Incorporate new mail.
Meta-C	Commit changes.
Meta-R	Rescan folder.
Meta-P	Pack folder.
Meta-S	Sort folder.
Meta-space	View next message.
Meta-c	Mark copy.
Meta-d	Mark deleted.
Meta-f	Forward the selected or current message.
Meta-m	Mark move.
Meta-n	View next message.
Meta-p	View previous message.
Meta-r	Reply to the selected or current message.
Meta-u	Unmark.
Control-V	Scroll the table of contents forward.
Meta-V	Scroll the table of contents backward.
Control-v	Scroll the view forward.
Meta-v	Scroll the view backward.

Text Editing Commands

All of the text editing commands are actually defined by the Text widget in the Athena widget set. The commands may be bound to different keys than the defaults described below through the X Toolkit Intrinsic key re-binding mechanisms. See Chapter 11, *Setting Resources*, and Appendix G, *Widget Resources*, for more details.

Whenever you are asked to enter any text, you will be using a standard text editing interface. Various control and meta keystroke combinations are bound to a somewhat Emacs-like set of commands. In addition, the pointer buttons may be used to select a portion of text or to move the insertion point in the text. Pressing pointer button 1 causes the insertion point to move to the pointer. Double-clicking button 1 selects a word, triple-clicking selects a line, quadruple-clicking selects a paragraph, and quintuple-clicking selects everything. Any selection may be extended in either direction by using pointer button 3.

In the following, a *line* refers to one displayed row of characters in the window, while a *paragraph* refers to the text between carriage returns. Text within a paragraph is broken into lines for display based on the current width of the window. When a message is sent, text is broken

into lines based upon the values of the `SendBreakWidth` and `SendWidth` application-specific resources.

The following keystroke combinations are defined:

Control-a	Move to the beginning of the current line.
Control-b	Move backward one character.
Control-d	Delete the next character.
Control-e	Move to the end of the current line.
Control-f	Move forward one character.
Control-h, or Backspace	Delete the previous character.
Control-j	New line and indent.
Control-k	Kill the rest of the current line. (Does not kill the carriage return at the end of the line. To do so, use Control-k twice. However, be aware that the second kill overwrites the text line in the kill buffer.)
Control-l	Refresh the window.
Control-m, Return, or Linefeed	New line.
Control-n	Move down to the next line.
Control-o	Divide this line into two lines at this point and move the cursor back up.
Control-p	Move up to the previous line.
Control-r	Search and replace backward.
Control-s	Search and replace forward.
Control-t	Transpose characters. (Swap the characters immediately before and after the cursor.)
Control-u	Perform next command four times. For example, the sequence Control-u, Control-n moves the cursor down four lines.
Control-v	Move down to the next screenful of text.
Control-w	Kill the selected text.
Control-y	Insert the last killed text. (If the last killed text is a carriage return—see Control-k above—a blank line is inserted.)
Control-z	Scroll the text up one line.
Meta-b	Move backward one word.

Meta-d	Delete the next word.
Meta-D	Kill the next word.
Meta-f	Move forward one word.
Meta-h, Meta-Backspace, or Meta-Delete Meta-H, Meta-Shift-Backspace, or Meta-Shift-Delete	Delete the previous word.
Meta-i	Insert a file. A dialog box will appear in which you can type the desired filename.
Meta-k	Kill to end of paragraph.
Meta-q	Join lines to form a paragraph.
Meta-v	Move up to the previous screenful of text.
Meta-y	Insert the last selected text here. Note that this can be text selected in some other text subwindow. Also, if you select some text in an <i>xterm</i> window, it may be inserted in an <i>xmh</i> window with this command. Pressing pointer button 2 is equivalent to this command.
Meta-z	Scroll one line down.
Meta-<	Move to the beginning of the file.
Meta->	Move to the end of the file.
Meta-]	Move forward one paragraph.
Meta-[Move backward one paragraph.

In addition, the pointer may be used to copy and paste text:

Button 1 Down	Start selection.
Button 1 Motion	Adjust selection.
Button 1 Up	End selection (copy).
Button 2 Down	Insert current selection (paste).
Button 3 Down	Extend current selection.

X Interface to mh**xmh** (continued)

Button 3 Motion	Adjust selection.
Button 3 Up	End selection (copy).

Confirmation Dialog Boxes

Whenever you press a button that may cause you to lose some work or is otherwise dangerous, a popup dialog box will appear asking you to confirm the action. This window will contain an Abort or No button and a Confirm or Yes button. Pressing the No button cancels the operation, and pressing Yes will proceed with the operation.

Some dialog boxes contain messages from *mh*. Occasionally when the message is more than one line long, not all of the text will be visible. Clicking on the message field will cause the dialog box to resize so that you can read the entire message.

Message Sequences

An *mh* message sequence is just a set of messages associated with some name. They are local to a particular folder; two different folders can have sequences with the same name. The sequence "all" is predefined in every folder; it consists of the set of all messages in that folder. As many as nine sequences may be defined for each folder, including the predefined "all" sequence. (The sequence "cur" is also usually defined for every folder; it consists of only the current message. *xmh* hides "cur" from the user, instead placing a "+" by the current message. Also, *xmh* does not support *mh*'s "unseen" sequence, so that one is also hidden from the user.)

The message sequences for a folder (including one for "all") are displayed in the Sequence menu, below the sequence commands. The table of contents (also known as the "toc") is at any one time displaying one message sequence. This is called the "viewed sequence," and its name will be displayed in the toc titlebar after the folder name. Also, at any time one of the sequences in the menu will have a check mark next to it. This is called the "selected sequence." Note that the viewed sequence and the selected sequence are not necessarily the same. (This all pretty much corresponds to the way folders work.)

The Open Sequence, Add to Sequence, Remove from Sequence, and Delete Sequence commands are active only if the viewed folder contains message-sequences other than the "all" sequence.

Note that none of the above actually affect whether a message is in the folder. Remember that a sequence is a set of messages within the folder; the above operations just affect what messages are in that set.

To create a new sequence, select the Pick menu entry. A new window will appear, with lots of places to enter text. Basically, you can describe the sequence's initial set of messages based on characteristics of the message. Thus, you can define a sequence to be all the messages that were from a particular person, or with a particular subject, and so on. You can also connect things up with Boolean operators, so you can select all things from "weissman" with a subject containing "xmh."

The layout is fairly obvious. The simplest cases are the easiest: just point to the proper field and type. If you enter in more than one field, it will only select messages which match all non-empty fields.

The more complicated cases arise when you want things that match one field or another one, but not necessarily both. That's what all the OR buttons are for. If you want all things with subjects that include "xmh" or "xterm," just press the OR button next to the Subject: field. Another box will appear where you can enter another subject.

If you want all things either from "weissman" or with subject "xmh," but not necessarily both, select the -Or- button. This will essentially double the size of the form. You can then enter weissman in a from: box on the top half, and "xmh" in a subject: box on the lower part.

If you select the Skip button, then only those messages that *don't* match the fields on that row are included.

Finally, several more boxes will appear in the bottom part of the window. One is the name of the sequence you're defining. (It defaults to the name of the selected sequence when Pick was pressed, or to "temp" if "all" was the selected sequence.) Another box defines which sequence to look through for potential members of this sequence; it defaults to the viewed sequence when Pick was pressed.

Two more boxes define a date range; only messages within that date range will be considered. These dates must be entered in RFC 822-style format: each date is of the form dd mmm yy hh:mm:ss zzz, where dd is a one or two digit day of the month, mmm is the three-letter abbreviation for a month, and yy is a year. The remaining fields are optional: hh, mm, and ss specify a time of day, and zzz selects a time zone. Note that if the time is left out, it defaults to midnight; thus if you select a range of "7 nov 86" – "8 nov 86", you will get only messages from the 7th, as all messages on the 8th will have arrived after midnight.

Date field specifies which field in the header to look at for this date range; it defaults to Date. If the sequence you're defining already exists, you can optionally merge the old set with the new; that's what the Yes and No buttons are all about. Finally, you can OK the whole thing, or Cancel it.

In general, most people will rarely use these features. However, it's nice to occasionally use Pick to find some messages, look through them, and then hit Delete Sequence to put things back in their original state.

Widget Hierarchy

In order to specify resources, it is useful to know the hierarchy of widgets which compose *xmh*. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name. The application class name is Xmh.

The hierarchy of the main toc and view window is identical for additional toc and view windows, except that a TopLevelShell widget is inserted in the hierarchy between the application shell and the Paned widget.

```

Xmh xmh
  Paned xmh
    SimpleMenu folderMenu
      SmeBSB open
      SmeBSB openInNew
      SmeBSB create
      SmeBSB delete
      SmeLine line
      SmeBSB close
    SimpleMenu tocMenu
      SmeBSB inc
      SmeBSB commit
      SmeBSB pack
      SmeBSB sort
      SmeBSB rescan
    SimpleMenu messageMenu
      SmeBSB compose
      SmeBSB next
      SmeBSB prev
      SmeBSB delete
      SmeBSB move
      SmeBSB copy
      SmeBSB unmark
      SmeBSB viewNew
      SmeBSB reply
      SmeBSB forward
      SmeBSB useAsComp
      SmeBSB print
    SimpleMenu sequenceMenu
      SmeBSB pick
      SmeBSB openSeq
      SmeBSB addToSeq
      SmeBSB removeFromSeq
      SmeBSB deleteSeq
      SmeLine line
      SmeBSB all
    SimpleMenu viewMenu
      SmeBSB reply
      SmeBSB forward
      SmeBSB useAsComp
      SmeBSB edit
      SmeBSB save
      SmeBSB print
    SimpleMenu optionMenu
      SmeBSB reverse
  Viewport.Core menuBox.clip
    Box menuBox
      MenuButton folderButton

```

```

                MenuButton tocButton
                MenuButton messageButton
                MenuButton sequenceButton
                MenuButton viewButton
                MenuButton optionButton
Grip grip
Label folderTitlebar
Grip grip
Viewport.Core folders.clip
    Box folders
        MenuButton inbox
        MenuButton drafts
            SimpleMenu menu
                SmeBSB <folder_name>
                .
                .
                .
Grip grip
Label tocTitlebar
Grip grip
Text toc
    Scrollbar vScrollbar
Grip grip
Label viewTitlebar
Grip grip
Text view
    Scrollbar vScrollbar
    Scrollbar hScrollbar

```

The hierarchy of the Create Folder popup dialog box:

```

TransientShell prompt
    Dialog dialog
        Label label
        Text value
        Command okay
        Command cancel

```

The hierarchy of the Notice dialog box, which reports messages from mh:

```

TransientShell notice
    Dialog dialog
        Label label
        Text value
        Command confirm

```

The hierarchy of the Confirmation dialog box:

```

TransientShell confirm
  Dialog dialog
    Label label
    Command yes
    Command no

```

The hierarchy of the dialog box which reports errors:

```

TransientShell error
  Dialog dialog
    Label label
    Command OK

```

The hierarchy of the composition window:

```

TopLevelShell xmh
  Paned xmh
    Label composeTitlebar
    Text comp
    Viewport.Core compButtons.clip
      Box compButtons
        Command close
        Command send
        Command reset
        Command compose
        Command save
        Command insert

```

The hierarchy of the view window:

```

TopLevelShell xmh
  Paned xmh
    Label viewTitlebar
    Text view
    Viewport.Core viewButtons.clip
      Box viewButtons
        Command close
        Command reply
        Command forward
        Command useAsComp
        Command edit
        Command save
        Command print

```

*The hierarchy of the pick window:
(Unnamed widgets have no name.)*


```

TopLevelShell xmh
  Paned xmh
    Label pickTitlebar
    Viewport.Core pick.clip
      Form form
        Form groupform

```

The first 6 rows of the pick window have identical structure:

```

      Form rowform
        Toggle
        Toggle
        Label
        Text
        Command

      Form rowform
        Toggle
        Toggle
        Text
        Text
        Command

      Form rowform
        Command

Viewport.Core pick.clip
  Form form
    Form groupform
      Form rowform
        Label
        Text
        Label
        Text

      Form rowform
        Label
        Text
        Label
        Text
        Label
        Text

      Form rowform
        Label
        Toggle
        Toggle

      Form rowform
        Command
        Command

```

See Appendix G, *Widget Resources* for a list of resources that can be set for the Athena widgets.

Application-specific Resources

The application class name is `Xmh`. Application-specific resource class names always begin with an uppercase character, but unless noted, are otherwise identical to the instance names.

Any of these resources may also be specified on the command line by using the X Toolkit Intrinsics resource specification mechanism. Thus, to run `xmh` showing all message headers,

```
% xmh -xrm '*HideBoringHeaders:off'
```

If `TocGeometry`, `ViewGeometry`, `CompGeometry`, or `PickGeometry` are not specified, then the value of `Geometry` is used instead. If the resulting height is not specified (e.g., "", "=500", "+0-0"), then the default height of windows is calculated from fonts and line counts. If the width is not specified (e.g., "", "=x300", "-0+0"), then half of the display width is used. If unspecified, the height of a pick window defaults to half the height of the display.

The following resources are defined:

banner

A short string that is the default label of the folder, Table of Contents, and view. The default is:

```
xmh    MIT X Consortium    R5
```

blockEventsOnBusy

Whether to disallow user input and show a busy cursor while `xmh` is busy processing a command. Default is `true`.

busyCursor

The name of the symbol used to represent the position of the pointer, displayed if `blockEventsOnBusy` is `true`, when `xmh` is processing a time-consuming command. The default is `watch`.

busyPointerColor

The foreground color of the busy cursor. Default is `XtDefaultForeground`.

checkFrequency

How often to check for new mail, make checkpoints, and rescan the Table of Contents, in minutes. If `checkNewMail` is `true`, `xmh` checks to see if you have new mail each interval. If `makeCheckpoints` is `true`, checkpoints are made every fifth interval. Also every fifth interval, the Table of Contents is checked for inconsistencies with the file system, and rescanned. To prevent all of these checks from occurring, set `checkFrequency` to 0. The default is 1. This resource is retained for backward compatibility with user resource files; see also `checkpointInterval`, `mailInterval`, and `rescanInterval`.

checkNewMail

If `true`, `xmh` will check at regular intervals to see if new mail has arrived for any of the top level folders. A visual indication will be given if new mail is waiting to be retrieved. Default is `true`. (See "Bugs.") The interval can be adjusted with the `checkFrequency` resource.

commandButtonCount

The number of command buttons to create in a buttonbox in between the toc and the view areas of the main window. *xmh* will create these buttons with the names *button1*, *button2* and so on, in a box with the name *commandBox*. The user can specify labels and actions for the buttons in a private resource file; see the section on "Actions." The default is 0.

compGeometry

Initial geometry for windows containing compositions.

cursor

The name of the symbol used to represent the pointer. Default is *left_ptr*.

draftsFolder

The folder used for message drafts. Default is *drafts*.

geometry

Default geometry to use. Default is none.

hideBoringHeaders

If "on", then *xmh* will attempt to skip uninteresting header lines within messages by scrolling them off. Default is on.

initialFolder

Which folder to display on startup. Can also be set with the command-line option *-initial*. Default is *inbox*.

initialIncFile

The file name of your incoming mail drop. *xmh* tries to construct a filename for the *inc -file* command, but in some installations (e.g., those using the Post Office Protocol) no file is appropriate. In this case, *initialIncFile* should be specified as the empty string, and *inc* will be invoked without a *-file* argument. The default is to use the value of the environment variable *MAIL*, or if that is not set, to append the value of the environment variable *USER* to */usr/spool/mail/*.

mailPath

The full path prefix for locating your mail folders. May also be set with the command-line option, *-path*. The default is the Path component in *\$HOME/.mh_profile*, or *\$HOME/Mail* if none.

mailWaitingFlag

If *true*, *xmh* will attempt to set an indication in its icon when new mail is waiting to be retrieved. If this option is *true*, then *checkNewMail* is assumed to be *true* as well. The *-flag* command-line option is a quick way to turn *mailWaitingFlag* on.

makeCheckpoints

If *true*, *xmh* will attempt to save checkpoints of volatile information. The frequency of checkpointing is controlled by the resource *checkFrequency*.

mhPath

The directory in which to find the *mh* commands. If a command isn't found here, then the directories in the user's path are searched. Default is */usr/local/mh6*.

pickGeometry

Initial geometry for pick windows.

pointerColor

The foreground color of the pointer. Default is *XtDefaultForeground*.

prefixWmAndIconName

Whether to prefix the window and icon name with "xmh:". Default is *true*.

printCommand

The *sh* command to execute to print a message. Note that standard output and standard error must be specifically redirected! If a message or range of messages is selected for printing, the full file path of each message file is appended to the specified print command. The default is *enscript >/dev/null 2>/dev/null*.

replyInsertFilter

A shell command to be executed when the Insert button is activated in a composition window. The full path and filename of the source message is added to the end of the command before being passed to *sh(1)*. The default filter is *cat*; i.e., it inserts the entire message into the composition. Interesting filters are: *awk -e '{print " "\$0}'* or *<mh directory>/lib/mhl -form mhl.body*.

reverseReadOrder

When *true*, the next message will be the message prior to the current message in the Table of Contents, and the previous message will be the message after the current message in the Table of Contents. The default is *false*.

sendBreakWidth

When a message is sent from *xmh*, lines longer than this value will be split into multiple lines, each of which is no longer than *sendWidth*. This value may be overridden for a single message by inserting an additional line in the message header of the form *sendBreakWidth: value*. This line will be removed from the header before the message is sent. The default is 85.

sendWidth

When a message is sent from *xmh*, lines longer than *sendBreakWidth* characters will be split into multiple lines, each of which is no longer than this value. This value may be overridden for a single message by inserting an additional line in the message header of the form *sendWidth: value*. This line will be removed from the header before the message is sent. The default is 72.

skipCopied

Whether to skip over messages marked for copying when using View Next Message and View Previous Message. Default is *true*.

skipDeleted

Whether to skip over messages marked for deletion when using View Next Message and View Previous Message. Default is `true`.

skipMoved

Whether to skip over messages marked for moving to other folders when using View Next Message and View Previous Message. Default is `true`.

stickyMenu

If `true`, when popup command menus are used, the most recently selected entry will be under the cursor when the menu pops up. Default is `false`. See the file *clients/xmh/Xmh.sample* for an example of how to specify resources for pop up command menus.

tempDir

Directory for *xmh* to store temporary directories. For privacy, a user might want to change this to a private directory. Default is */tmp*.

tocGeometry

Initial geometry for master *xmh* windows.

tocPercentage

The percentage of the main window that is used to display the Table of Contents. Default is 33.

tocWidth

How many characters to generate for each message in a folder's Table of Contents. Default is 100. Use 80 if you plan to use *mhl* a lot, because it will be faster, and the extra 20 characters may not be useful.

viewGeometry

Initial geometry for windows showing only a view of a message.

Actions

Because *xmh* provides action procedures which correspond to command functionality and installs accelerators, users can customize accelerators in a private resource file. *xmh* provides action procedures which correspond to entries in the command menus; these are given in the sections describing menu commands. For examples of specifying customized resources, see the file *clients/xmh/Xmh.sample*. Unpredictable results can occur if actions are bound to events or widgets for which they were not designed.

In addition to the actions corresponding to commands, these action routines are defined:

`XmhPushFolder([foldername, . . .])`

Pushes each of its argument(s) onto a stack of folder names. If no arguments are given, the selected folder is pushed onto the stack.

`XmhPopFolder()`

Pops one folder name from the stack and sets the selected folder.

XmhPopupFolderMenu()

Should always be taken when the user selects a folder button. A folder button represents a folder and zero or more subfolders. The menu of subfolders is built upon the first reference, by this routine. If there are no subfolders, this routine will mark the folder as having no subfolders, and no menu will be built. In that case, the menu button emulates a toggle button. When subfolders exist, the menu will popup, using the menu button action `PopupMenu()`.

XmhSetCurrentFolder()

Allows menu buttons to emulate toggle buttons in the function of selecting a folder. This action is for `Menubutton` widgets only, and sets the selected folder.

XmhLeaveFolderButton()

Ensures that the menu button behaves properly when the user moves the pointer out of the menu button window.

XmhPushSequence([sequencename, . . .])

Pushes each of its arguments onto the stack of sequence names. If no arguments are given, the selected sequence is pushed onto the stack.

XmhPopSequence()

Pops one sequence name from the stack of sequence names, which then becomes the selected sequence.

XmhPromptOkayAction()

Equivalent to pressing the *Okay button* in the *Create Folder* popup.

XmhCancelPick()

Equivalent to pressing the *Cancel button* in the *Pick* window.

Customization Using mh

The initial text displayed in a composition window is generated by executing the corresponding *mh* command; i.e., *comp*, *repl*, or *forw*, and therefore message components may be customized as specified for those commands. *comp* is executed only once per invocation of *xmh* and the message template is re-used for each successive new composition.

Files*~/.mh_profile*

mh profile, used if the MH environment variable is not set.

~/Mail

Directory of folders, used if the *mh* profile cannot be found.

~/xmhcheck

Optional, for multiple mail drops in cooperation with *slocal*.

/usr/local/mh6

mh commands, as a last resort (see `mhPath`).

~/Mail/<folder>/xmhcache

scan output in each folder.

~/Mail/<folder>/mh_sequences
Sequence definitions in each folder.

/tmp Temporary files (see `tempDir`).

See Also

X, `xrdb`, `mh(1)`; Appendix G, *Widget Resources*; the Nutshell Handbook *MH and xmh: E-mail for Users and Programmers*.

Bugs

When the user closes a window, all windows which are transient for that window should also be closed by *xmh*.

When `XmhUseAsComposition` and `XmhViewUseAsComposition` operate on messages in the `DraftsFolder`, *xmh* disallows editing of the composition if the same message is also being viewed in another window.

Occasionally after committing changes, the table of contents will appear to be completely blank when there are actually messages present. When this happens, refreshing the display, or typing Control-I in the table of contents, will often cause the correct listing to appear. If this doesn't work, force a rescan of the folder.

Should recognize and use the "unseen" message-sequence.

Should determine by itself if the user hasn't used *mh* before, and offer to create the *.mh_profile*, instead of hanging on *inc*.

A few commands are missing (rename folder, resend message).

`WM_DELETE_WINDOW` protocol doesn't work right when requesting deletion of the first toc and view, while trying to keep other *xmh* windows around.

Doesn't support annotations when replying to messages.

Doesn't allow folders to be shared without write permission.

Doesn't recognize private sequences.

mh will report that the *.mh_sequences* file is poorly formatted if any sequence definition in a particular folder contains more than *BUFSIZ* characters. *xmh* tries to capture these messages and display them when they occur, but it cannot correct the problem.

Copyright

Copyright 1988, 1989, Digital Equipment Corporation.
Copyright 1989, 1991, Massachusetts Institute of Technology.
See X for a full statement of rights and permissions.

X Interface to mh

xmh *(continued)*

Author

Terry Weissman, formerly of Digital Western Research Laboratory;
Donna Converse, MIT X Consortium.

Reference Pages

Name

xmodmap – keyboard and pointer modifier utility.

Syntax

xmodmap [*options*] [*filename*]

Description

xmodmap is a utility for displaying and editing the X keyboard *modifier map* and *keymap table* that client applications use to convert keycodes into keysyms. *xmodmap* is intended to be run from a user's X startup script to set up the keyboard according to personal tastes.

With no arguments, *xmodmap* displays the current map. See Chapter 14, *Setup Clients*, for more information.

Options

xmodmap accepts the following options:

-display [*host*]:*server*[.*screen*]

Specifies the name of the display to use. *host* is the hostname of the physical display, *server* specifies the display server number, and *screen* specifies the screen number. Either or both of the *host* and *screen* elements to the display specification can be omitted. If *host* is omitted, the local display is assumed. If *screen* is omitted, screen 0 is assumed (and the period is unnecessary). The colon and (display) *server* are necessary in all cases. For example:

```
% xmodmap -display your_node:0.0
```

specifies the screen 0 on server 0 on the display identified by *your_node*. If the host is omitted, the local display is assumed. If the screen is omitted, the screen 0 is assumed; the server and colon (:) are necessary in all cases.

The **-display** option can be abbreviated as **-d**, unless the client accepts another option that begins with "d."

-e *expression*

Specifies an expression to be executed. Any number of expressions may be specified from the command line.

-grammar

Indicates that a help message describing the expression grammar used in files and with **-e** expressions should be printed on the standard error.

-help Indicates that a brief description of the command-line arguments should be printed on the standard error. This will be done whenever an invalid argument is given to *xmodmap*.

-quiet

Turns off the verbose logging. This is the default.

Keyboard Modifier Utility

xmodmap (continued)

- n Indicates that *xmodmap* should not change the mappings, but should display what it would do, as *make(1)* does when given this option. (Cannot be used with expressions to change the pointer mapping.)
- pm Indicates that the current modifier map should be printed on the standard output.
- pk Indicates that the current keymap table should be printed on the standard output.
- pke Indicates that the current keymap table should be printed on the standard output in the form of expressions that can be fed back to *xmodmap*.
- pp Indicates that the current pointer map should be printed on the standard output.
- A lone dash means that the standard input should be used as the input file.
- verbose Indicates that *xmodmap* should print logging information as it parses its input.

The *filename* argument specifies a file containing *xmodmap* expressions to be executed. This file is usually kept in the user's home directory and has a name like *.xmodmaprc*.

Expression Grammar

The *xmodmap* program reads a list of expressions and parses them all before attempting to execute any of them. This makes it possible to refer to keysyms that are being redefined in a natural way without having to worry as much about name conflicts. Allowable expressions include:

keycode *NUMBER = KEYSYMNAME . . .*

The list of keysyms is assigned to the indicated keycode (which may be specified in decimal, hex, or octal and can be determined by running the *xev* program in the examples directory).

keysym *KEYSYMNAME = KEYSYMNAME . . .*

The *KEYSYMNAME* on the left hand side is translated into matching keycodes used to perform the corresponding set of *keycode* expressions. The list of keysym names may be found in the header file *<X11/keysymdef.h>* (without the *XK_* prefix) or the keysym database */usr/lib/X11/XKkeysymDB*. Note that if the same keysym is bound to multiple keys, the expression is executed for each matching keycode.

clear *MODIFIERNAME*

This removes all entries in the modifier map for the given modifier, where valid names are: *Shift*, *Lock*, *Control*, *Mod1*, *Mod2*, *Mod3*, *Mod4*, and *Mod5* (case does not matter in modifier names, although it does matter for all other names). For example, *clear Lock* will remove any keys that were bound to the lock modifier.

add *MODIFIERNAME = KEYSYMNAME . . .*

This adds all the keys having the given keysyms to the indicated modifier map. The keysym names are evaluated after all input expressions are read to make it easy to write expressions to swap keys. (See the "Examples" section.)

```
remove MODIFIERNAME = KEYSYMNAME . . .
```

This removes all the keys having the given keysyms from the indicated modifier map. Unlike add, the keysym names are evaluated as the line is read in. This allows you to remove keys from a modifier without having to worry about whether or not they have been reassigned.

```
pointer = default
```

This sets the pointer map back to its default settings (button 1 generates a code of 1, button 2 generates a 2, etc.).

```
pointer = N1 N2 N3
```

This sets the pointer map to contain the button codes *N1*, *N2*, and *N3*, where *N1*, *N2* and *N3* are numbers. The list always starts with the first physical button.

Lines that begin with an exclamation mark (!) are taken as comments.

If you want to change the binding of a modifier key, you must also remove it from the appropriate modifier map.

Examples

Many pointers are designed such that the first button is pressed using the index finger of the right hand. People who are left handed frequently find that it is more comfortable to reverse the button codes that get generated so that the primary button is pressed using the index finger of the left hand. This could be done on a 3-button pointer as follows:

```
% xmodmap -e "pointer = 3 2 1"
```

Many editor applications support the notion of Meta keys (similar to Control keys except that Meta is held down instead of Control). However, some servers do not have a Meta keysym in the default keymap table, so one needs to be added by hand. The following command will attach Meta to the Multi-language key (sometimes labeled Compose Character). It also takes advantage of the fact that applications that need a Meta key need simply to get the keycode and don't require the keysym to be in the first column of the keymap table. This means that applications that are looking for a Multi_key (including the default modifier map) won't notice any change.

```
% xmodmap -e "keysym Multi_key = Multi_key Meta_L"
```

One of the more simple, yet convenient, uses of *xmodmap* is to set the keyboard's "rubout" key to generate an alternate keysym. This frequently involves exchanging Backspace with Delete to be more comfortable to the user. If the *ttymodes* resource in *xterm* is set as well, all terminal emulator windows will use the same key for erasing characters:

```
% xmodmap -e "keysym BackSpace = Delete"
% echo "XTerm*ttymodes: erase ^?" | xrdp -merge
```

Some keyboards do not automatically generate less than and greater than characters when the comma and period keys are shifted. This can be remedied with *xmodmap* by resetting the bindings for the comma and period with the following scripts:

```
!
! make shift-, be < and shift-. be >
!
keysym comma = comma less
keysym period = period greater
```

One of the more irritating differences between keyboards is the location of the Control and Shift Lock keys. A common use of *xmodmap* is to swap these two keys as follows:

```
!
! Swap Caps_Lock and Control_L
!
remove Lock = Caps_Lock
remove Control = Control_L
keysym Control_L = Caps_Lock
keysym Caps_Lock = Control_L
add Lock = Caps_Lock
add Control = Control_L
```

The *keycode* expression is useful for assigning the same *keysym* to multiple keycodes. Although unportable, it also makes it possible to write scripts that can reset the keyboard to a known state. The following script sets the Backspace key to generate Delete (as shown above), flushes all existing caps lock bindings, makes the CapsLock key a control key, makes F5 generate Escape, and makes Break/Reset function as a shift lock.

```
!
! On the HP, the following keycodes have key caps as listed:
!
!      101  Backspace
!      55   Caps
!      14   Ctrl
!      15   Break/Reset
!      86   Stop
!      89   F5
!
keycode 101 = Delete
keycode 55 = Control_R
clear Lock
add Control = Control_R
```

```
keycode 89 = Escape  
keycode 15 = Caps_Lock  
add Lock = Caps_Lock
```

See Also

X, xev; Chapter 14, *Setup Clients*.

Bugs

Every time a `keycode` expression is evaluated, the server generates a `MappingNotify` event on every client. This can cause some thrashing. All of the changes should be batched together and done at once. Clients that receive keyboard input and ignore `MappingNotify` events will not notice any changes made to keyboard mappings.

`xmodmap` should generate `add` and `remove` expressions automatically whenever a `keycode` that is already bound to a modifier is changed.

There should be a way to have the `remove` expression accept `keycodes` as well as `keysyms` for those times when you really mess up your mappings.

Authors

Rewritten by Jim Fulton, MIT X Consortium, from an earlier version by David Rosenthal of Sun Microsystems.

Name

`xpr` – print an X window dump.

Syntax

`xpr` [*options*] [*filename*]

Description

`xpr` takes as input a window dump file produced by `xwd` and formats it for output on PostScript printers, the DEC LN03 or LA100, the IBM PP3812 page printer, the HP LaserJet (or other PCL printers), or the HP PaintJet. If you do not supply a filename, standard input is used. By default, `xpr` prints the largest possible representation of the window on the output page. Options allow you to add headers and trailers, specify margins, adjust the scale and orientation, and append multiple window dumps to a single output file. Output is sent to standard output unless you specify `-output filename`. See Chapter 8, *Other Clients*, for some examples of usage.

Options

`xpr` accepts the following options:

`-append filename`

Specifies a filename previously produced by `xpr` to which the window contents are to be appended.

`-compact`

Compresses white pixels on PostScript printers.

`-cutoff level`

Changes the intensity level where colors are mapped to either black or white for monochrome output on a LaserJet printer. The *level* is expressed as a percentage of full brightness. Fractions are allowed.

`-density dpi`

Indicates what dot-per-inch density should be used by the HP printer.

`-device printer_device`

Specifies the device on which the file is to be printed. Currently the following printers are supported:

`ln03` Digital LN03.

`la100` Digital LA100.

`ljet` HP LaserJet series and other monochrome PCL devices, such as ThinkJet, QuietJet, RuggedWriter, HP2560 series, and HP2930 series printers.

`pjet` HP PaintJet (color mode).

`pjetxl` HP PaintJet XL Color Graphics Printer (color mode).

pp IBM PP3812.

ps PostScript printer.

As of Release 5, the default is ps (PostScript). (In prior releases, the default printer is the LN03.) `-device lw` (Apple LaserWriter) is equivalent to `-device ps` and is provided only for backwards compatibility.

`-gamma correction`

Changes the intensity of the colors printed by PaintJet XL printers. The *correction* is a floating point value in the range 0.00 to 3.00. Consult the operator's manual to determine the correct value for the specific printer.

`-gray 2 | 3 | 4`

Uses a simple 2×2 , 3×3 , or 4×4 grey scale conversion on a color image, rather than mapping to strictly black and white. This doubles, triples, or quadruples the effective width and height of the image. `-gray` is not supported for HP and IBM printers.

`-header header`

Specifies a header string to be printed above the window. Default is no header.

`-height inches`

Specifies the maximum height of the page.

`-landscape`

Prints the window in landscape mode. By default, a window is printed such that its longest side follows the long side of the paper.

`-left inches`

Specifies the left margin in inches. Fractions are allowed. By default, the window is centered on the page.

`-noff` When specified in conjunction with `-append`, the window appears on the same page as the previous window.

`-noposition`

Causes header, trailer, and image positioning command generation to be bypassed for LaserJet, PaintJet, and PaintJet XL printers.

`-output filename`

Specifies an output filename. If this option is not specified, standard output is used.

`-plane number`

Specifies which bit plane to use in an image. The default is to use the entire image and map values into black and white based on color intensities.

`-portrait`

Prints the window in portrait mode. By default, a window is printed such that its longest side follows the long side of the paper.

`-psfig`

Suppress translation of the PostScript picture to the center of the page.

- render *algorithm*
Allows PaintJet XL printers to render the image with the best quality versus performance tradeoff. Consult the operator's manual to determine which *algorithms* are available.
- rv Reverses the foreground and background colors.
- scale *scale*
Affects the size of the window on the page. The PostScript, LN03, and HP printers are able to translate each bit in a window pixel map into a grid of a specified size. For example, each bit might translate into a 3 × 3 grid. This is specified by `-scale 3`. By default, a window is printed with the largest scale that fits onto the page for the specified orientation.
- slide
Allows overhead transparencies to be printed using the PaintJet and PaintJet XL printers.
- split *n*
Allows you to split a window onto several pages. This might be necessary for large windows that would otherwise cause the printer to overload and print the page in an obscure manner.
- top *inches*
Specifies the top margin for the picture in inches. Fractions are allowed. By default, the window is centered on the page.
- trailer *trailer*
Specifies a trailer string to be printed below the window. Default is no trailer.
- width *inches*
Specifies the maximum width of the page.

Limitations

The current version of *xpr* can generally print out on the LN03 most X windows that are not larger than two-thirds of the screen. For example, it will be able to print out a large *emacs* window, but it will usually fail when trying to print out the entire screen. The LN03 has memory limitations that can cause it to incorrectly print very large or complex windows. The two most common errors encountered are "band too complex" and "page memory exceeded." In the first case, a window may have a particular band (a row six pixels deep) that contains too many changes (from black to white to black). This will cause the printer to drop part of the line and, possibly, parts of the rest of the page. The printer will flash the number "1" on its front panel when this problem occurs. A possible solution to this problem is to increase the scale of the picture, or to split the picture onto two or more pages. The second problem, "page memory exceeded," will occur if the picture contains too much black, or if the picture contains complex half-tones such as the background color of a display. When this problem occurs the printer will automatically split the picture into two or more pages. It may flash the number "5"

on its front panel. There is no easy solution to this problem. It will probably be necessary to either cut and paste, or rework the application to produce a less complex picture.

There are several limitations on the use of *xpr* with the LA100: the picture will always be printed in portrait mode, there is no scaling, and the aspect ratio will be slightly off.

Support for PostScript output currently cannot handle the `-append`, `-noff`, or `-split` options.

The `-compact` option is supported *only* for PostScript output. It compresses white space but not black space, so it is not useful for reverse-video windows.

For color images, should map directly to PostScript image support.

HP Printer Specifics

If no `-density` is specified on the command line, 300 dots per inch will be assumed for `ljet` and 90 dots per inch for `pjet`. Allowable *density* values for a LaserJet printer are 300, 150, 100, and 75 dots per inch. Consult the operator's manual to determine densities supported by other printers.

If no `-scale` is specified, the image will be expanded to fit the printable page area.

The default printable page area is 8 × 10.5 inches. Other paper sizes can be accommodated using the `-height` and `-width` options.

Note that a 1024 × 768 image fits the default printable area when processed at 100 dpi with `scale=1`; the same image can also be printed using 300 dpi with `scale=3`, but will require that considerably more data be transferred to the printer.

xpr may be tailored for use with monochrome PCL printers other than the LaserJet. To print on a ThinkJet (HP2225A), *xpr* could be invoked as:

```
% xpr -density 96 -width 6.667 filename
```

or for black-and-white output to a PaintJet:

```
% xpr -density 180 filename
```

The monochrome intensity of a pixel is computed as $0.30 \cdot R + 0.59 \cdot G + 0.11 \cdot B$. If a pixel's computed intensity is less than the `-cutoff` level, it will print as white. This maps light-on-dark display images to black-on-white hardcopy. The default cutoff intensity is 50% of full brightness. Example: specifying `-cutoff 87.5` moves the white/black intensity point to 87.5% of full brightness.

A LaserJet printer must be configured with sufficient memory to handle the image. For a full page at 300 dots per inch, approximately 2MB of printer memory is required.

Color images are produced on the PaintJet at 90 dots per inch. The PaintJet is limited to 16 colors from its 330 color palette on each horizontal print line. *xpr* will issue a warning message if more than 16 colors are encountered on a line. *xpr* will program the PaintJet for the first 16 colors encountered on each line and use the nearest matching programmed value for other colors present on the line.

Print X Window Dump

xpr (continued)

Specifying the `-rv` (reverse video) option for the PaintJet will cause black and white to be interchanged on the output image. No other colors are changed.

Multiplane images must be recorded by `xwd` in ZPixmap format. Single plane (monochrome) images may be in either XYPixmap or ZPixmap format.

Some PCL printers do not recognize image positioning commands. Output for these printers will not be centered on the page and header and trailer strings may not appear where expected.

The `-gamma` and `-render` options are supported only on the PaintJet XL printers.

The `-slide` option is not supported for LaserJet printers.

The `-split` option is not supported for HP printers.

See Also

`xwd`, `xdpr`, `xwud`, X; Chapter 8, *Other Clients*.

Copyright

Copyright 1988, Massachusetts Institute of Technology.

Copyright 1986, Marvin Solomon and the University of Wisconsin.

Copyright 1988, Hewlett-Packard Company.

See X for a full statement of rights and permissions.

Authors

Michael R. Gretzinger, MIT Project Athena;

Jose Capo, MIT Project Athena (PP3812 support);

Marvin Solomon, University of Wisconsin;

Bob Scheifler, MIT;

Angela Bock and E. Mike Durbin, Rich Inc. (greyscale);

Larry Rupp, Hewlett-Packard (HP printer support).

Name

`xprop` – display window and font properties for X.

Syntax

`xprop` [*options*]

Description

The *xprop* utility displays window and font properties in an X server. One window or font is selected using the command line arguments or, in the case of a window, by clicking on the desired window. A list of properties is then given, possibly with formatting information.

For each of these properties, its value on the selected window or font is printed using the supplied formatting information, if any. If no formatting information is supplied, internal defaults are used. If a property is not defined on the selected window or font, “not defined” is printed as the value for that property. If no property list is given, all the properties possessed by the selected window or font are printed.

A window may be selected in one of four ways. First, if the desired window is the root window, the `-root` option may be used. If the desired window is not the root window, it may be selected in two ways on the command line, either by ID number such as might be obtained from *xwininfo*, or by name if the window possesses a name. The `-id` option selects a window by ID number in either decimal or hex (must start with 0x) while the `-name` option selects a window by name.

The last way to select a window does not involve the command line at all. If none of `-font`, `-id`, `-name`, and `-root` is specified, a crosshair cursor is displayed and the user is allowed to choose any visible window by pressing any pointer button in the desired window. If it is desired to display properties of a font as opposed to a window, the `-font` option must be used.

Other than the above four options, the `-help` option for obtaining help, and the `-grammar` option for listing the full grammar for the command line, all the other command-line options are used in specifying both the format of the properties to be displayed and how to display them. The `-len n` option specifies that at most *n* bytes of any given property will be read and displayed. This is useful, for example, when displaying the cut buffer on the root window, which could run to several pages if displayed in full.

Normally each property name is displayed by printing first the property name, then its type (if it has one) in parentheses, followed by its value. The `-notype` option specifies that property types should not be displayed. The `-fs` option is used to specify a file containing a list of formats for properties, while the `-f` option is used to specify the format for one property.

The formatting information for a property actually consists of two parts, a *format* and a *dformat*. The *format* specifies the actual formatting of the property (i.e., is it made up of words, bytes, or longs, etc.), while the *dformat* specifies how the property should be displayed.

The following paragraphs describe how to construct *formats* and *dformats*. However, for the vast majority of users and uses, this should not be necessary as the built-in defaults contain

the *formats* and *dformats* necessary to display all the standard properties. It should be necessary to specify *formats* and *dformats* only if a new property is being dealt with or the user dislikes the standard display format. New users especially are encouraged to skip this part.

A *format* consists of one of 0, 8, 16, or 32 followed by a sequence of one or more format characters. The 0, 8, 16, or 32 specifies how many bits per field there are in the property. Zero is a special case, meaning use the field size information associated with the property itself. (This is needed only for special cases like type INTEGER, which is actually three different types depending on the size of the fields of the property.)

A value of 8 means that the property is a sequence of bytes, while a value of 16 means that the property is a sequence of words. The difference between these two lies in the fact that the sequence of words will be byte swapped, while the sequence of bytes will not be, when read by a machine of the opposite byte order of the machine that originally wrote the property. For more information on how properties are formatted and stored, consult Volume One, *Xlib Programming Manual*.

Once the size of the fields has been specified, it is necessary to specify the type of each field (i.e., is it an integer, a string, an atom, or what?). This is done using one format character per field. If there are more fields in the property than format characters supplied, the last character will be repeated as many times as necessary for the extra fields. The format characters and their meanings are as follows:

- a The field holds an atom number. A field of this type should be of size 32.
- b The field is a Boolean. A 0 means `False` while anything else means `True`.
- c The field is an unsigned number, a cardinal.
- i The field is a signed integer.
- m The field is a set of bit flags, 1 meaning on.
- s This field and the next ones, until either a 0 or the end of the property, represent a sequence of bytes. This format character is usable only with a field size of 8 and is most often used to represent a string.
- x The field is a hex number (like `c` but displayed in hex—most useful for displaying window IDs and the like).

An example *format* is `32ica`, which is the format for a property of three fields of 32 bits each, the first holding a signed integer, the second an unsigned integer, and the third an atom.

The format of a *dformat* (unlike that of a *format*) is not so rigid. The only limitations on a *dformat* is that it may not start with a letter or a dash. This is so that it can be distinguished from a property name or an option. A *dformat* is a text string containing special characters instructing that various fields be printed at various points in a manner similar to the formatting string used by *printf*. For example, the *dformat* “`is ($0, $1 \)\n`” would render the POINT 3, -4 which has a *format* of `32ii` as “`is (3, -4)\n`”.

Any character other than a \$, ?, \, or a (in a *dformat* prints as itself. To print out one of \$, ?, \, or (, precede it by a \. For example, to print out a \$, use \\$. Several special backslash sequences are provided as shortcuts. \n will cause a newline to be displayed while \t will cause a tab to be displayed. \o, where o is an octal number, will display character number o.

A \$ followed by a number *n* causes field number *n* to be displayed. The format of the displayed field depends on the formatting character used to describe it in the corresponding *format*. That is, if a cardinal is described by c, it will print in decimal while if it is described by a x it will be displayed in hex.

If the field is not present in the property (this is possible with some properties), <field not available> is displayed instead. \$*n*+ will display field number *n*, then a comma, then field number *n*+1, then another comma, then ... until the last field defined. If field *n* is not defined, nothing is displayed. This is useful for a property that is a list of values.

A ? is used to start a conditional expression, a kind of if-then statement. ?*exp*(*text*) will display *text* if and only if *exp* evaluates to non-zero. This is useful for two things. First, it allows fields to be displayed if and only if a flag is set. And second, it allows a value such as a state number to be displayed as a name rather than just as a number. The syntax of *exp* is as follows:

$$\begin{aligned} \text{exp} &::= \text{term} \mid \text{term}=\text{exp} \mid !\text{exp} \\ \text{term} &::= n \mid \$n \mid mn \end{aligned}$$

The ! operator is a logical “not,” changing 0 to 1 and any non-zero value to 0. = is an equality operator. Note that internally all expressions are evaluated as 32-bit numbers, so -1 is not equal to 65535. = returns 1 if the two values are equal and 0 if not. *n* represents the constant value *n*, while \$*n* represents the value of field number *n*. *mn* is 1 if flag number *n* in the first field having format character *m* in the corresponding *format* is 1, 0 otherwise.

Examples: ?m3(count: \$3\n) displays field 3 with a label of count if and only if flag number 3 (count starts at 0!) is on. ?\$2=0(True)?!\$2=0(False) displays the inverted value of field 2 as a Boolean.

In order to display a property, *xprop* needs both a *format* and a *dformat*. Before *xprop* uses its default values of a *format* of 32x and a *dformat* of “ = { \$0+ }\n”, it searches several places in an attempt to find more specific formats. First, a search is made using the name of the property. If this fails, a search is made using the type of the property. This allows type STRING to be defined with one set of formats while allowing property WM_NAME, which is of type STRING, to be defined with a different format. In this way, the display formats for a given type can be overridden for specific properties.

The locations searched are in order: the format, if any, specified with the property name (as in 8x WM_NAME), the formats defined by -f options in last to first order, the contents of the file specified by the -fs option, if any, the contents of the file specified by the environment variable XPROPFORMATS, if any, and finally *xprop*'s built-in file of formats.

The format of the files referred to by the `-fs` option and the `XPROPFORMATS` variable is one or more lines of the following form:

```
name format [dformat]
```

Where *name* is either the name of a property or the name of a type, *format* is the *format* to be used with *name*, and *dformat* is the *dformat* to be used with *name*. If *dformat* is not present, “ = \$0+\n” is assumed.

Options

xprop accepts the following options:

`-display [host]:server[.screen]`

Allows you to specify the display and server to connect to. *host* is the hostname of the physical display, *server* specifies the display server number, and *screen* specifies the screen number. For example:

```
% xprop -display your_node:0.1
```

specifies screen 1 on server 0 on the display named by *your_node*. Either or both of the *host* and *screen* elements to the display specification can be omitted. If *host* is omitted, the local display is assumed. If *screen* is omitted, screen 0 is assumed (and the period is unnecessary). The colon and (display) *server* are necessary in all cases.

`-f name format [dformat]`

Specifies that the format for *name* should be *format* and that the *dformat* for *name* should be *dformat*. If *dformat* is missing, “ = \$0+\n” is assumed.

`-font font`

Allows the user to specify that the properties of font *font* should be displayed.

`-frame`

Specifies that, when selecting a window by hand (i.e., if none of `-name`, `-root`, or `-id` are given), *xprop* should look at the window manager frame (if any) instead of looking for the client window.

`-fs file`

Specifies that file *file* should be used as a source of more formats for properties.

`-grammar`

Prints out a detailed grammar for all command-line options.

`-help` Prints out a summary of command-line options.

`-id id`

Allows the user to select window *id* on the command line rather than using the pointer to select the target window. This is very useful in debugging X applications where the target window is not mapped to the screen or where the use of the pointer might be impossible or interfere with the application.

- len *n*
Specifies that at most *n* bytes of any property should be read or displayed.
- name *name*
Allows the user to specify on the command line that the window named *name* is the target window, rather than using the pointer to select the target window.
- notype
Specifies that the type of each property should not be displayed.
- remove *property_name*
Specifies the name of a property to be removed from the indicated window.
- root
Specifies that X's root window is the target window. This is useful in situations where the root window is completely obscured.
- spy
Indicates that *xprop* should examine window properties forever, looking for property change events.

Examples

To display the name of the root window: `prop -root WM_NAME`

To display the window manager hints for the clock: `xprop -name xclock WM_HINTS`

To display the start of the cut buffer: `xprop -root -len 100 CUT_BUFFER0`

To display the point size of the fixed font: `xprop -font fixed POINT_SIZE`

To display all the properties of window # 0x200007: `xprop -id 0x200007`

Environment Variables

XPROPFORMATS

Specifies the name of a file from which additional formats are to be obtained.

See Also

X, `xwininfo`.

Author

Mark Lillibridge, MIT Project Athena.

Name

xrdb – X server resource database utility.

Syntax

```
xrdb [options] [filename | -]
```

Description

xrdb is used to get or set the contents of the RESOURCE_MANAGER property on the root window of screen 0, or the SCREEN_RESOURCES property on the root window of any or all screens, or everything combined. You would normally run this program from your X startup file. Chapter 11, *Setting Resources*, describes how to use *xrdb*.

Most X clients use the RESOURCE_MANAGER and SCREEN_RESOURCES properties to get user preferences about color, fonts, and so on for applications. Having this information in the server (where it is available to all clients) instead of on disk solves the problem in previous versions of X that required you to maintain *defaults* files on every machine that you might use. It also allows for dynamic changing of defaults without editing files.

The RESOURCE_MANAGER property is used for resources that apply to all screens of the display. The SCREEN_RESOURCES property on each screen specifies additional (or overriding) resources to be used for that screen. (When there is only one screen, SCREEN_RESOURCES is normally not used, all resources are just placed in the RESOURCE_MANAGER property.)

The *filename* (or the standard input if – or no input file is given) is optionally passed through the C preprocessor with the being used:

BITS_PER_RGB=number

The number of significant bits in an RGB color specification. This is the log base 2 of the number of distinct shades of each primary that the hardware can generate. Note that it is usually not related to the number of PLANES.

CLASS=visualclass

One of StaticGray, GrayScale, StaticColor, PseudoColor, TrueColor, or DirectColor. This is the visual class of the root window of the default screen.

CLIENTHOST=hostname

The name of the host on which *xrdb* is running.

COLOR Defined only if CLASS is one of StaticColor, PseudoColor, or DirectColor.

HEIGHT=number

The height of the default screen in pixels.

PLANES=*number*

The number of bit planes (the depth) of the root window of the default screen. Defined only if CLASS is one of StaticColor, PseudoColor, TrueColor, or DirectColor.

RELEASE=*number*

The vendor release number for the server. The interpretation of this number will vary depending on VENDOR.

REVISION=*number*

The X protocol minor version supported by this server (currently 0).

SERVERHOST=*hostname*

HOST=*hostname*

The hostname portion of the display to which you are connected.

VENDOR=*number*

A string specifying the vendor of the server.

VERSION=*number*

The X protocol major version supported by this server (should always be 11).

WIDTH=*number*

The width of the default screen in pixels.

X_RESOLUTION=*number*

The x resolution of the default screen in pixels per meter.

Y_RESOLUTION=*number*

The y resolution of the default screen in pixels per meter.

Lines that begin with an exclamation mark (!) are ignored and may be used as comments.

Note that since *xrdb* can read from standard input, it can be used to change the contents of properties directly from a terminal or from a shell script.

Options

xrdb accepts the following options:

- all This option indicates that operation should be performed on the screen-independent resource property (RESOURCE_MANAGER), as well as the screen-specific property (SCREEN_RESOURCES) on every screen of the display. For example, when used in conjunction with *-query*, the contents of all properties are output. For *-load* and *-merge*, the input file is processed once for each screen. The resources which occur in common in the output for every screen are collected, and these are applied as the screen-independent resources. The remaining resources are applied for each individual per-screen property. This the default mode of operation. (Available as of Release 5.)

- backup** *string*
Specifies a suffix to be appended to the filename used with **-edit** to generate a backup file.
- cpp** *filename*
Specifies the pathname of the C preprocessor program to be used. Although *xrdb* was designed to use *cpp*, any program that acts as a filter and accepts the **-D**, **-I**, and **-U** options may be used.
- display** [*host*]:*server*[.*screen*]
Specifies the X display server to be used; see X. It also specifies the screen to use for the **-screen** option and it specifies the screen from which preprocessor symbols are derived for the **-global** option.
- Dname**[=*value*]
Is passed through to the preprocessor and is used to define symbols for use with conditionals such as *#ifdef*.
- edit** *filename*
Indicates that the contents of the specified properties should be edited into the given file, replacing any values already listed there. This allows you to put changes that you have made to your defaults back into your resource file, preserving any comments or preprocessor lines.
- global**
This option indicates that the operation should only be performed on the SCREEN_RESOURCES property of the default screen of the display. (Available as of Release 5.)
- help** This option (or any unsupported option) will cause a brief description of the allowable options and parameters to be printed.
- Idirectory**
Is passed through to the preprocessor and is used to specify a directory to search for files that are referenced with *#include*.
- load** Indicates that the input should be loaded as the new value of the properties, replacing whatever was there (i.e., the old contents are removed). This is the default action.
- merge**
Indicates that the input should be merged with, instead of replacing, the current contents of the specified properties. Note that this option does a lexicographic sorted merge of the two inputs, which is almost certainly not what you want, but remains for backward compatibility.
- n** Indicates that changes to the specified properties (when used with **-load** or **-merge**) or to the resource file (when used with **-edit**) should be shown on the standard output, but should not be performed.

- nocpp**
Indicates that *xrdb* should not run the input file through a preprocessor before loading it into the properties.
- query**
Indicates that the current contents of the specified properties should be printed onto the standard output. Note that since preprocessor commands in the input resource file are part of the input file, not part of the property, they won't appear in the output from this option. The **-edit** option can be used to merge the contents of the properties back into the input resource file without damaging preprocessor commands.
- quiet**
Indicates that warning about duplicate entries should not be displayed.
- remove**
Indicates that the specified properties should be removed from the server.
- retain**
Indicates that the server should be instructed not to reset if *xrdb* is the first client. This is never necessary under normal conditions, since *xdm* and *xinit* always act as the first client.
- screen**
This option indicates that the operation should only be performed on the `SCREEN_RESOURCES` property of the default screen of the display. (Available as of Release 5.)
- screens**
This option indicates that the operation should be performed on the `SCREEN_RESOURCES` property of each screen of the display. For **-load** and **-merge**, the input file is processed for each screen. (Available as of Release 5.)
- symbols**
Indicates that the symbols that are defined for the preprocessor should be printed onto the standard output.
- Uname**
Is passed through to the preprocessor and is used to remove any definitions of this symbol.

Files

Generalizes `~/.Xdefaults` files.

See Also

X; Chapter 11, *Setting Resources*.

Bugs

The default for no arguments should be to query, not to overwrite, so that it is consistent with other programs.

Authors

Bob Scheifler, Phil Karlton, rewritten from the original by Jim Gettys. Copyright 1991, Digital Equipment Corporation and MIT.

Name

xrefresh – refresh all or part of an X screen.

Syntax

xrefresh [*options*]

Description

xrefresh is a simple X program that causes all or part of your screen to be repainted. This is useful when system messages have displayed on your screen. *xrefresh* maps a window on top of the desired area of the screen and then immediately unmaps it, causing refresh events to be sent to all applications. By default, a window with no background is used, causing all applications to repaint “smoothly.” However, the various options can be used to indicate that a solid background (of any color) or the root window background should be used instead.

See Chapter 8, *Other Clients*, for more information about *xrefresh*. In certain cases, you can run the *xconsole* client to prevent system messages from obscuring the screen. See the *xconsole* reference page and Appendix A, *Managing Your Environment*, for more information.

Options

xrefresh accepts the following options:

-black

Use a black background (in effect, turning off all of the electron guns to the tube). This can be somewhat disorienting as everything goes black for a moment.

-display [*host*]:*server*[.*screen*]

Allows you to specify the display, server and screen to refresh. *host* is the hostname of the physical display, *server* specifies the display server number, and *screen* specifies the screen number.

% **xrefresh -display** *your_node*:0.1

specifies screen 1 of server 0 on the display named by *your_node*. Either or both of the *host* and *screen* elements to the display specification can be omitted. If *host* is omitted, the local display is assumed. If *screen* is omitted, screen 0 is assumed (and the period is unnecessary). The colon and (display) *server* are necessary in all cases.

-geometry *geometry*

Specifies the portion of the screen to be repainted. (This is generally pointless.)

The **-geometry** option can be (and often is) abbreviated to **-g**, unless there is a conflicting option that begins with “g”. The argument to the geometry option (*geometry*) is referred to as a “standard geometry string,” and has the form *widthx-height±xoff±yoff*.

-none This is the default. All of the windows simply repaint.

-root Use the root window background.

Refresh X Screen

xrefresh *(continued)*

`-solid color`

Use a solid background of the specified color. Try green.

`-white`

Use a white background. The screen just appears to flash quickly, and then repaints.

Resources

The *xrefresh* program uses the routine *XGetDefault(3X)* to read defaults, so its resource names are all capitalized.

Black, White, Solid, None, Root

Determines what sort of window background to use.

Geometry

Determines the area to refresh. Not very useful.

See Also

X, *xconsole*; Chapter 8, *Other Clients*; Appendix A, *Managing Your Environment*.

Bugs

It should have just one default type for the background.

Author

Jim Gettys, Digital Equipment Corp., MIT Project Athena.

Name

xscdd – Xcms property builder.

Syntax

```
xscdd <inputfile> outputfile
```

Description

xscdd is a program developed by Tektronix, Inc., to create a color database for the Xcms Color Management System. *xscdd* takes as standard input the file produced by *xcrtca* and produces on standard output a file containing the property data loaded onto the root window by *xcmsdb*. The *contrib/clients/xcrtca/monitors* directory contains *xcrtca* output files and *xcmsdb* input files created from *xscdd* for several machines used at the X Consortium.

Caveats

This program has been coded for a Sun SparcStation and for a default visual with 8 bits_per_rgb.

See Also

xcrtca, *xcmsdb*; Chapter 12, *Specifying Color*.

Author

Dave Sternlicht, Keith Packard, MIT X Consortium;
Al Tabayoyon, Chuck Adams, Tektronix Inc.

Name

`xset` – user preference utility for X.

Syntax

`xset` [*options*]

Description

`xset` allows you to set various preferences for the display, pointer, and keyboard. Generally, all settings are reset to their defaults when you log out. However, in certain cases, settings specified for a particular display may be carried over from session to session. For example, some X terminals can be configured to retain settings between logins. Regardless of the environment, it's generally a good idea to run `xset` from the user's startup file. See Chapter 14, *Setup Clients*, for more information.

Options

`xset` accepts the following options. (Note that not all X implementations are guaranteed to honor all of these options.)

- b Controls bell volume, pitch, and duration. The `b` option accepts up to three numerical parameters (*volume*, *pitch*, and *duration*), a preceding dash (`-`), or an `on/off` flag. If no parameters are given, or the `on` flag is used, the system defaults will be used. If the dash or `off` are given, the bell will be turned off. If only one numerical parameter is given, the bell *volume* will be set to that value, as a percentage of its maximum. Likewise, the second numerical parameter specifies the bell *pitch*, in hertz, and the third numerical parameter specifies the *duration* in milliseconds. Note that not all hardware can vary the bell characteristics. The X server will set the characteristics of the bell as closely as it can to the user's specifications.

`-bc`, `bc`

Controls *bug compatibility* mode in the server, if possible. The option with a preceding dash (`-`) disables the mode; the option alone enables the mode.

The need for this option is determined by the following circumstances. Various pre-R4 clients pass illegal values in some protocol requests, and pre-R4 servers do not correctly generate errors in these cases. Such clients, when run with an R4 server, will terminate abnormally or otherwise fail to operate correctly. Bug compatibility mode explicitly reintroduces certain bugs into the X server, so that many such clients can still be run.

This mode should be used with care; new application development should be done with this mode disabled. Be aware that the server must support the MIT-SUNDRY-NONSTANDARD protocol extension in order for this option to work.

- c Controls key click. The `c` option can take an optional value, a preceding dash (`-`), or an `on/off` flag. If no parameter or the `on` flag is given, the system defaults will be used. If the dash or `off` flag is used, the keyclick will be disabled. If a value from 0

to 100 is given, it is used to indicate volume, as a percentage of the maximum. The X server will set the volume to the nearest value that the hardware can support.

`-display [host]:server[.screen]`

Allows you to specify the host, server, and screen for which to set preferences. *host* is the hostname of the physical display, *server* specifies the server number, and *screen* specifies the screen number. For example,

```
% xset -display your_node:0.1
```

specifies screen 1 of server 0 on the display named by *your_node*. Either or both of the *host* and *screen* elements to the display specification can be omitted. If *host* is omitted, the local display is assumed. If *screen* is omitted, screen 0 is assumed (and the period is unnecessary). The colon and *server* are necessary in all cases.

`fp= path`

Sets the font path used by the server. *path* must be a directory or a comma-separated list of directories. The directories are interpreted by the server, not the client, and are server-dependent. (Directories that do not contain font databases created by *mkfontdir* will be ignored by the server.)

`fp default`

Restores the default font path.

`fp rehash`

Causes the server to reread the font databases in the current font path. This is generally used only when adding new fonts to a font directory (after running *mkfontdir* to recreate the font database).

`-fp path` or `fp- path`

The `-fp` and `fp-` options remove elements from the current font path. *path* must be a directory or comma-separated list of directories.

`+fp path` or `fp+ path`

The `+fp` and `fp+` options prepend and append, respectively, elements to the current font path. *path* must be a directory or a comma-separated list of directories.

`led`

Controls the turning on or off of one or all of the LEDs. The `led` option accepts an optional integer, a preceding dash (`-`) or an `on/off` flag. If no parameter or the `on` flag is given, all LEDs are enabled. If a preceding dash or the flag `off` is given, all LEDs are disabled. If a value between 1 and 32 is given, that LED will be enabled or disabled, depending on the existence of a preceding dash. A common LED that can be controlled is the Caps Lock LED. `xset led 3` enables LED #3. `xset -led 3` disables it. The particular LED values may refer to different LEDs on different hardware.

`m`

Controls the mouse parameters. The acceleration can be specified as an integer or as a fraction (with the numerator and denominator separated by a slash, for example, `1/2`). The parameters for the mouse are *acceleration* and *threshold*. The mouse, or whatever pointer is connected to the machine, will go *acceleration* times as fast

when it travels more than *threshold* pixels in a short time. This way, the mouse can be used for precise alignment when it is moved slowly, yet it can be set to travel across the screen in a flick of the wrist when desired. One or both parameters for the *m* option can be omitted, but if only one is given, it will be interpreted as the acceleration. If no parameters or the flag *default* is used, the system defaults will be set.

- p** Controls pixel color values. The parameters are the color map entry number in decimal, and a color specification. The root background colors may be changed on some servers by altering the entries for `BlackPixel` and `WhitePixel`. Although these are often 0 and 1, they need not be. Also, a server may choose to allocate those colors privately, in which case an error will be generated. The map entry must not be a read-only color, or an error will result.
- q** Gives you information on the current settings.
- r** Controls the autorepeat. If a preceding dash or the *off* flag is used, autorepeat will be disabled. If no parameters or the *on* flag is used, autorepeat will be enabled.
- s** Controls the screen saver parameters. The *s* option accepts up to two numerical parameters (*time* and *cycle*), a *blank/noblank* flag, an *expose/noexpose* flag, an *on/off* flag, or the *default* flag. If no parameters or the *default* flag is used, the system will be set to its default screen saver characteristics. The *on/off* flags simply turn the screen saver functions on or off. The *blank* flag sets the preference to blank the video (if the hardware can do so) rather than display a background pattern, while *noblank* sets the preference to display a pattern rather than blank the video. The *expose* flag sets the preference to allow window exposures (the server can freely discard window contents), while *noexpose* sets the preference to disable the screen saver unless the server can regenerate the screens without causing exposure events. The *time* and *cycle* parameters for the screen saver function determine how long the server must be inactive for screen saving to activate, and the period to change the background pattern to avoid burn in, respectively. The arguments are specified in seconds. If only one numerical parameter is given, it will be used for the *time*.

See Also

X, Xserver, xmodmap, xrdb, xsetroot; Chapter 14, *Setup Clients*.

Authors

Bob Scheifler, MIT Laboratory for Computer Science;
David Krikorian, MIT Project Athena (X11 version).

xsetroot

Set Root Window Characteristics—

Name

xsetroot – root window parameter setting utility.

Syntax

xsetroot [*options*]

Description

xsetroot allows you to tailor the appearance of the root (background) window on a display. You can experiment with *xsetroot* until you find a look that you like, then put the *xsetroot* command that produces it into your X startup file. If you do not specify any options or you specify `-def`, the window is reset to its defaults. The `-def` option can be specified along with other options and only the non-specified characteristics will be reset to the default state. See Chapter 14, *Setup Clients*, for instructions on using *xsetroot*.

Options

xsetroot accepts the following options. Note that only one of the background color/tile changing options (`-solid`, `-gray`, `-gray`, `-bitmap`, or `-mod`) can be specified at a time. *color* can be specified as a color name or a numeric value. See Chapter 12, *Specifying Color*, for more information.

`-bg color`

Sets the background color of the root window. Foreground and background colors are meaningful only in combination with `-cursor`, `-bitmap`, or `-mod`. The default is white.

`-bitmap filename`

Uses the bitmap specified in the file to set the window pattern. The entire background is made up of repeated tiles of the bitmap. You can make your own bitmap files using the *bitmap* client or you can use those available with X, usually found in the directory `/usr/include/X11/bitmaps`. The default is a gray mesh.

`-cursor cursorfile maskfile`

Specifies the cursor shape to use as the root window pointer. The *cursorfile* and *maskfile* are bitmaps, which can be made with the *bitmap* client. (Refer to Chapter 7, *Graphics Utilities*, for more information on creating bitmaps.) The mask file may need to be all black until you are accustomed to the way masks work. The default root window pointer is an X cursor.

`-cursor_name standard_cursor_name`

Changes the root window cursor to one of the standard cursors from the cursor font. (See Appendix D, *Standard Cursors*, for a list and pictures.) To specify a cursor name as an argument to a command-line option, the `XC_` prefix must be stripped from the name.

`-def` Resets unspecified attributes to the default values. (Restores the background to the gray mesh background and the pointer to the hollow X pointer.) If you specify `-def` and other options, only the non-specified options are reset to their defaults.

Set Root Window Characteristics

xsetroot (continued)**-display** [*host*]:*server*[.*screen*]

Allows you to specify the host, server, and screen of the root window. *host* is the hostname of the physical display, *server* specifies the server number, and *screen* specifies the screen number. For example,

```
% xsetroot -display your_node:0.1
```

specifies screen 1 of server 0 on the display named by *your_node*. Either or both of the *host* and *screen* elements to the display specification can be omitted. If *host* is omitted, the local display is assumed. If *screen* is omitted, screen 0 is assumed (and the period is unnecessary). The colon and (display) *server* are necessary in all cases.

-fg *color*

Sets the foreground color of the root window. Foreground and background colors are only meaningful in combination with **-cursor**, **-bitmap**, or **-mod**. The default is black.

-gray or **-grey**

Creates a gray background.

-help Displays a brief description of the allowable options.**-mod** *x y*

Makes a plaid-like grid pattern on your screen. *x* and *y* are integers ranging from 1 to 16 and are used to determine the dimensions in pixels of the plaid rectangles. Try some different combinations. Zero and negative numbers are taken as 1.

-name *string*

Sets the name of the background window to *string*. There is no default value. This option allows a client to refer to the root window by name. (Usually, a name is assigned to a window so that the window manager can use a text representation when the window is converted to an icon. However, since the root window cannot be iconified, this function does not apply.)

-rv Reverses the foreground color and the background color when used with another option, such as **-mod**. (Normally the foreground color is black and the background color is white.) Without another specified option, **-rv** returns the root (background) window to the default state.**-solid** *color*

Sets the root window color. This option is primarily useful on color servers. The default color is a gray mesh.

See Also

X, **xset**, **xrdb**; Chapter 14, *Setup Clients*; Chapter 12, *Specifying Color*.

xsetroot *(continued)*

Set Root Window Characteristics

Author

Mark Lillibridge, MIT Project Athena.

Name

xstdcmap – X standard colormap utility.

Syntax

```
xstdcmap [options]
```

Description

The *xstdcmap* utility can be used to selectively define standard colormap properties. It is intended to be run from a user's X startup script to create standard colormap definitions in order to facilitate sharing of scarce colormap resources among clients. Where at all possible, colormaps are created with read-only allocations.

Options

xstdcmap accepts the following options:

- all Specifies that all six standard colormap properties should be defined on each screen of the display. Not all screens will support visuals under which all six standard colormap properties are meaningful. *xstdcmap* will determine the best allocations and visuals for the colormap properties of a screen. Any previously existing standard colormap properties will be replaced.
- best Specifies that the RGB_BEST_MAP should be defined.
- blue Specifies that the RGB_BLUE_MAP should be defined.
- default Specifies that the RGB_DEFAULT_MAP should be defined.
- delete *map* Specifies that a standard colormap property should be removed. *map* may be one of: default, best, red, green, blue, or grey.
- display [*host*]:*server*[.*screen*] Allows you to specify the display, server, and screen to connect to. *host* is the host-name of the physical display, *server* specifies the server number, and *screen* specifies the screen number. For example:


```
xstdcmap -display your_node:0.1
```

 specifies screen 1 of server 0 on the display named by *your_node*. Either or both the *host* and *screen* elements can be omitted. If *host* is omitted, the local display is assumed. If *screen* is omitted, screen 0 is assumed (and the period is unnecessary). The colon and *server* are necessary in all cases.
- green Specifies that the RGB_GREEN_MAP should be defined.
- grey Specifies that the RGB_GRAY_MAP should be defined.

xstdcmap *(continued)*

Define Standard Colormaps

- help Specifies that a brief description of the command-line arguments should be printed on the standard error. This will be done whenever an unhandled argument is given to *xstdcmap*.
- red Specifies that the RGB_RED_MAP should be defined.
- verbose Specifies that *xstdcmap* should print logging information as it parses its input and defines the standard colormap properties.

See Also

X.

Author

Donna Converse, MIT X Consortium.

Name

xterm – window terminal emulator.

Syntax

xterm [*options*]

Description

The *xterm* program is a terminal emulator for the X Window System. Chapter 5 of this guide explains how to work effectively using *xterm*. This reference page provides some of the information necessary to customize *xterm*. See Part Two of this guide for general instructions on customizing X clients.

The *xterm* client provides DEC VT102 and Tektronix 4014 compatible terminals for programs that can't use the window system directly. If the underlying operating system supports terminal resizing capabilities (for example, the SIGWINCH signal in systems derived from BSD 4.3), *xterm* will use the facilities to notify programs running in the window whenever it is resized.

The VT102 and Tektronix 4014 terminals each have their own window so that you can edit text in one and look at graphics in the other at the same time. To maintain the correct aspect ratio (height/width), Tektronix graphics will be restricted to the largest box with a 4014's aspect ratio that will fit in the window. This box is located in the upper-left area of the window.

Although both windows can be displayed at the same time, one of them is considered the *active* window for receiving keyboard input and terminal output. This is the window that contains the text cursor. The active window can be chosen through escape sequences, the VT Options menu in the VT102 window, and the Tek Options menu in the 4014 window.

The Release 5 version of *xterm* provides four menus that allow you to manage the VT102 and Tektronix windows: Main Options, VT Options, Tek Options, and VT Fonts.

xterm automatically highlights the text cursor when the pointer enters the window (selected) and unhighlights it when the pointer leaves the window (unselected). If the window is the focus window, then the text cursor is highlighted no matter where the pointer is.

In VT102 mode, there are escape sequences to activate and deactivate an alternate screen buffer, which is the same size as the display area of the window. When activated, the current screen is saved and replaced with the alternate screen. Saving of lines scrolled off the top of the window is disabled until the normal screen is restored. The *termcap(5)* entry for *xterm* allows the visual editor *vi* to switch to the alternate screen for editing and to restore the screen on exit.

In either VT102 or Tektronix mode, there are escape sequences to change the name of the windows and to specify a new log file name. See Appendix E, *xterm Control Sequences*, for details. Enabling the escape sequence to change the log file name is a compile-time option; by default this escape sequence is ignored for security reasons.

Options

xterm accepts all of the standard X Toolkit command-line options, which are listed on the X reference page. (We've included some of the more commonly used Toolkit options later in this section.)

In addition, *xterm* accepts the following application-specific options. Note that if the option begins with a + instead of a -, the option is restored to its default value. (Specifying the default with *+option* can be useful for overriding the opposite value in an *.Xresources* file or other prior resource specification.)

-help

Causes *xterm* to print out a verbose message describing its options.

-132 Causes the VT102 DECCOLM escape sequence, which switches between 80- and 132-column mode, to be recognized, enabling the *xterm* window to resize properly. By default, the DECCOLM escape sequence is ignored. (See Appendix C for more information on *xterm* escape sequences.)

(This option can be turned on and off from the *xterm* VT Options menu, described below.)

-ah/+ah

-ah specifies that *xterm* should *always* highlight the text cursor. By default, *xterm* will display a highlighted text cursor only when a window has the input focus and a hollow text cursor when the focus is elsewhere. **+ah** specifies the default.

-aw/+aw

-aw specifies that auto-wraparound of text should be allowed. This allows the cursor to automatically wrap to the beginning of the next line when it is at the right-most position on a line and text is output. This is the default. **+aw** specifies that auto-wraparound should not be allowed. (Available as of Release 5.)

-b *innerborder*

Specifies the width of the inner border (the distance between the outer edge of the characters and the window border) in pixels. The default is two pixels.

-C

Specifies that the *xterm* window should receive console output. This is not supported on all systems. To obtain console output, you must be the owner of the console device, and you must have read and write permission for it. If you are running X under *xdm* on the console screen you may need to have the session startup and reset programs explicitly change the ownership of the console device in order to get this option to work. Running the *xconsole* client is generally preferable to *xterm -C*. See Appendix A, *Managing Your Environment* and the *xconsole* reference page for more information.

-cb, +cb

-cb specifies that triple-clicking to select a line does not include the newline at the end of the line. The default is to include the newline. **+cb** specifies the default. (Available as of Release 5.)

- cc *characterclassrange:value[,...]*
Sets classes indicated by the given ranges for use in selecting by words. See “Specifying Character Classes” below.
- cn, +cn
-cn indicates that newlines should not be cut in line mode selections; +cn indicates that newlines should be cut in line mode selections.
- cr *color*
Specifies the color to use for the text cursor. The default is to use the same foreground color that is used for text.
- cu, +cu
-cu enables the *curses* fix. Several programs that use the *curses(3x)* cursor motion package have some difficulties with VT102-compatible terminals. The bug occurs when you run the *more* program on a file containing a line that is exactly the width of the window and that is followed by a line beginning with a tab. The leading tabs are not displayed. This option causes the tabs to be displayed correctly.

+cu indicates that *xterm* should not work around this *curses* bug.

(This option can also be turned on and off from the VT Options menu, described below.)
- e *command [arguments]*
Specifies the command (and its arguments) to be run in the *xterm* window. It also sets the window title and icon name to be the name of the program being executed if neither -T nor -n are given on the command line. The -e option, command, and the arguments must appear last on the *xterm* command line; for example, *xterm -rv -e more bigfile &*.
- fb *font*
Uses the specified font as the bold font. This font must be the same height and width as the normal font. If only one of the normal or bold fonts is specified, it is used as the normal font and the bold font is produced by overstriking this font. The default is to overstrike the normal font.
- im, +im
-im forces the use of insert mode by adding appropriate entries to the TERMCAP environment variable. This is useful if the system termcap is broken. The default is not to force insert mode. +im specifies the default. (Available as of Release 5.)
- j, +j
-j indicates that *xterm* should do jump scrolling. Normally, text is scrolled one line at a time; this option allows *xterm* to move multiple lines at a time so that it doesn't fall as far behind. The use of jump scrolling is strongly recommended since it makes *xterm* much faster when scanning through large amounts of text. The VT100 escape sequences for enabling and disabling smooth scroll and the Enable Jump Scroll item of the VT Options menu can also be used to toggle this feature.

The +j option specifies that *xterm* not do jump scrolling.

-j specifies the default behavior.

(This option can be turned on and off from the VT Options menu, described below.)

-l, +l

-l logs *xterm* input/output into a file called *XtermLog.xxxx*, where *xxxx* represents the process ID number. To display your data, turn off logging using the *xterm* menu, then type `cat XtermLog.xxxx` at the *xterm* window prompt and the output file is sent to your *xterm* window. Logging allows you to keep track of the sequence of data and is particularly helpful while debugging code.

+l specifies that *xterm* not do logging.

(This option can also be turned on and off from the VT Options menu, described below.)

-lf *file*.

Specifies the file to which the data is written rather than the default *XtermLog.xxxx*, where *xxxx* is the process identification of *xterm* (the file is created in the directory in which *xterm* is started in or the home directory for a login *xterm*). If *file* begins with a "|," then the rest of the string is assumed to be a command to be executed by the shell and a pipe is opened to the process.

-ls, +ls

-ls indicates that the shell that is started in the *xterm* window be a login shell (i.e., the first character of `argv[0]` will be a dash, indicating to the shell that it should read the user's *.login* or *.profile*).

+ls indicates that the shell that is started should not be a login shell (i.e., it will be a normal "subshell").

-mb, +mb

-mb turns on the margin bell; the default is bell off. +mb indicates that the margin bell should not be rung.

(This option can also be turned on and off from the VT Options menu, described below.)

-mc *milliseconds*

Specifies the maximum time between multi-click selections.

-ms *color*

Sets the color of the pointer. The default is to use the foreground color.

-nb *number*

Sets the distance at which the margin bell rings for the right margin. Default is 10 characters.

-rw, +rw

-rw turns on the reverse-wraparound mode that allows the cursor to wrap around from the leftmost column to the rightmost column of the previous line. Allows you to backspace to the previous line and overstrike data or erase data with the spacebar.

+rw indicates that reverse-wraparound should not be enabled.

(This option can also be turned on and off from the VT Options menu, described below.)

-Sccn Specifies the last two letters of the name of a pseudo-terminal to use in slave mode, plus the number of the inherited file descriptor. The option is parsed “%c%c%d”. This allows *xterm* to be used as an input and output channel for an existing program and is sometimes used in specialized applications.

-s, +s **-s** allows *xterm* to scroll asynchronously with the display, meaning that the screen does not have to be kept completely up-to-date while scrolling. *xterm* saves data in memory which is displayed later. This allows *xterm* to run faster when network latencies are high and is useful when running *xterm* across a large internet or many gateways.

+s indicates that *xterm* should scroll synchronously.

-sb, +sb

-sb indicates that some number of lines that are scrolled off the top of the window should be saved and that a scrollbar should be displayed at startup so those lines can be viewed.

+sb indicates that a scrollbar should not be displayed at startup.

(This feature can also be turned on and off from the VT Options menu, described below.)

-sf, +sf

-sf indicates that the Sun function key escape codes should be generated for function keys; **+sb** indicates that the standard escape codes should be generated for function keys. This is the default.

-si, +si

-si disables repositioning the cursor at the bottom of the scroll region when the process sends output; **+si** indicates that the cursor should be repositioned at the bottom of the scroll region on output.

(This feature can also be turned on and off from the VT Options menu, described below.)

-sk, +sk

-sk causes the cursor to be repositioned at the bottom of the scroll region when a key is pressed; **+sk** indicates that pressing a key while using the scrollbar should not cause the cursor to be repositioned at the bottom of the scroll region.

(This feature can also be turned on and off from the VT Options menu, described below.)

-sl *number*

Specifies the maximum number of lines to be saved that are scrolled off the top of the window. Default is 64 lines.

-t, +t

-t causes the startup *xterm* window to be the Tektronix window rather than the VT102 window; **+t** causes the startup window to be the VT102 window. This is the default.

-tm *string*

Specifies a series of terminal-setting keywords followed by the characters that should be bound to those functions, similar to the *stty* program. Allowable keywords include: *intr*, *quit*, *erase*, *kill*, *eof*, *eol*, *swtch*, *start*, *stop*, *brk*, *susp*, *dsusp*, *rprnt*, *flush*, *weras*, and *lnext*. Control characters may be specified as *^char* (e.g., *^c* or *^u*), and *^?* may be used to indicate delete.

-tn *name*

Specifies the name of the terminal type to be set in the TERM environment variable. This terminal type must exist in the *termcap(5)* database and should have *li#* and *co#* entries.

-ut/+ut

-ut indicates that *xterm* shouldn't write a record into the the system log file */etc/utmp*.

+ut indicates that *xterm* should write a record into the system log file */etc/utmp*.

-vb/+vb

-vb causes your terminal window to flash whenever an event occurs that would ordinarily cause your terminal bell to ring.

+vb indicates that a visual bell should not be used.

(This feature can be turned on and off from the Main Options menu, described below.)

-wf/+wf

-wf indicates that *xterm* should wait for the window to be mapped the first time before starting the subprocess so that the initial terminal size settings and environment variables are correct. It is the application's responsibility to catch subsequent terminal size changes.

+wf indicates that *xterm* should not wait before starting the subprocess.

The following X Toolkit options are commonly used with *xterm*:

-bd *color*

Sets the color of the border. Default of the highlighted border is black. Default of the unhighlighted border is grey.

- bg *color*
Sets the background color of the *xterm* window. Default is white.
- bw *pixels*
Specifies the width of the *xterm* window border in pixels. Default is one pixel.
- display [*host*]:*server*[.*screen*]
Specifies the display, server, and screen on which to create the window. *host* is the hostname of the physical display, *server* specifies the server number, and *screen* specifies the screen number. For example:

```
% xterm -display your_node:0.1
```


specifies that an *xterm* be created on screen 1 of server 0 on the display named by *your_node*. Either or both of the *host* and *screen* elements to the display specification can be omitted. If *host* is omitted, the local display is assumed. If *screen* is omitted, screen 0 is assumed (and the period is unnecessary). The colon and *server* are necessary in all cases.
- fg *color*
Sets the color of the text (foreground). Default is black.
- fn *font*
Uses the specified font instead of the default font (*fixed*). You can use any fixed-width font.
- geometry *geometry*
xterm takes this geometry specification for the VT102 window. The `-geometry` option can be (and often is) abbreviated to `-g`, unless there is a conflicting option that begins with "g". The argument to the geometry option (*geometry*) is referred to as a "standard geometry string," and has the form *width**x**height**±**xoff**±**yoff*.
- iconic
Causes *xterm* to display an *xterm* icon rather than an *xterm* window when it starts up.
- name *app_name*
Specifies the application name under which resources are to be obtained, rather than the default executable filename. *app_name* should not contain "." or "*" characters.
- title *string*
Specifies the window title string, which may be displayed by window managers if the user so chooses. The default title is the command line specified after the `-e` option, if any, otherwise the application name.
- rv
Reverses the foreground and background colors.

(This option can be turned on and off from the VT Options menu, described below.)

-xrm resourcestring

Specifies a resource string to be used with this instance of the application. This is especially useful for setting resources that do not have command-line option equivalents.

The following command-line arguments are provided for compatibility with older versions (prior to Release 3). They may not be supported in the next release as the X Toolkit provides standard options that accomplish most of the same tasks.

%geometry

Specifies the preferred size and location of the Tektronix window. It is shorthand for specifying the `tekGeometry` resource.

#geometry

Specifies the preferred position of the icon. It is shorthand for specifying the `iconGeometry` resource. The width and height values of the geometry string are optional.

-n string

Specifies the icon name for the *xterm* window. It is shorthand for specifying the `*iconName` resource. Note that this is not equivalent to the Toolkit option `-name`. The default icon name is the name of a program run with the `-e` option, if any, otherwise the application name.

-r

Indicates that reverse video should be simulated by swapping the foreground and background colors. It is equivalent to `-rv`.

-T string

Specifies the title for the *xterm* window. It is equivalent to `-title`.

-w pixels

Specifies the width in pixels of the border surrounding the window. It is equivalent to `-bw`.

Resources

The program understands all of the Core resource names and classes as well as the following:

autoWrap (class AutoWrap)

Specifies whether or not auto-wraparound should be enabled. The default is `True`.

iconGeometry (class IconGeometry)

Specifies the preferred size and position of the application when iconified. It is not necessarily obeyed by all window managers.

iconName (class IconName)

Specifies the icon name. The default is the application name.

`useInsertMode` (class `UseInsertMode`)

If `true`, forces the use of insert mode by adding appropriate entries to the `TERMCAP` environment variable. This is useful if the system `termcap` is broken. The default is `False`. (Available as of Release 5.)

`termName` (class `TermName`)

Specifies the terminal type name to be set in the `TERM` environment variable.

`title` (class `Title`)

Specifies a string that may be used by the window manager (e.g., in a titlebar) when displaying this application.

`ttyModes` (class `TtyModes`)

Specifies a string containing terminal setting keywords and the characters to which they may be bound. Allowable keywords include: `intr`, `quit`, `erase`, `kill`, `eof`, `eol`, `swtch`, `start`, `stop`, `brk`, `susp`, `dsusp`, `rprnt`, `flush`, `weras`, and `lnext`. Control characters may be specified as `^char` (e.g., `^c` or `^u`), and `^?` may be used to indicate Delete. This is very useful for overriding the default terminal settings without having to do an `stty` every time an *xterm* is started.

`utmpInhibit` (class `UtmpInhibit`)

Specifies whether or not *xterm* should try to record the user's terminal in `/etc/utmp`.

`sunFunctionKeys` (class `SunFunctionKeys`)

Specifies whether or not Sun Function Key escape codes, instead of standard escape sequences, should be generated for function keys.

`waitForMap` (class `WaitForMap`)

Specifies whether or not *xterm* should wait for the initial window map before starting the subprocess. The default is `False`. (Available as of Release 5.)

The following resources are specified as part of the `vt100` widget (class `VT100`):

`allowSendEvents` (class `AllowSendEvents`)

Specifies whether or not synthetic key and button events (generated using the X protocol `SendEvent` request) should be interpreted or discarded. The default is `False` meaning they are discarded. Note that allowing such events creates a very large security hole.

`alwaysHighlight` (class `AlwaysHighlight`)

Specifies whether or not *xterm* should always display a highlighted text cursor. By default, a hollow text cursor is displayed whenever the pointer moves out of the window or the window loses the input focus.

`appcursorDefault` (class `AppcursorDefault`)

If `true`, the cursor keys are initially in application mode. The default is `False`. (Available as of Release 5.)

- `appkeypadDefault` (class `AppkeypadDefault`)
If true, the keypad keys are initially in application mode. The default is `False`.
(Available as of Release 5.)
- `autoWrap` (class `AutoWrap`)
Specifies whether or not auto-wraparound should be enabled. The default is `True`.
(Available as of Release 5.)
- `bellSuppressTime` (class `BellSuppressTime`)
Specifies number of milliseconds after a bell command is sent during which additional bells will be suppressed. Default is 200. If set nonzero, additional bells will also be suppressed until the server reports that processing of the first bell has been completed; this feature is most useful with the visible bell. (Available as of Release 5.)
- `boldFont` (class `Font`)
Specifies the name of the bold font to use instead of overstriking the normal font.
- `c132` (class `C132`)
Specifies whether or not the VT102 DECCOLM escape sequence should be honored. The default is `False`.
- `cutNewline` (class `CutNewline`)
If false, triple-clicking to select a line does not include the newline at the end of the line. If true, the newline is selected. The default is `True`. (Available as of Release 5.)
- `cutToBeginningOfLine` (class `CutToBeginningOfLine`)
If false, triple-clicking to select a line selects only from the current word forward. If true, the entire line is selected. The default is `True`. (Available as of Release 5.)
- `charClass` (class `CharClass`)
Specifies comma-separated lists of character class bindings of the form `[low-high:value]`. These are used in determining which sets of characters should be treated the same when doing cut and paste. See "Character Classes" below.
- `curses` (class `Curses`)
Specifies whether or not the last column bug in the cursor should be worked around. The default is `False`.
- `background` (class `Background`)
Specifies the color to use for the background of the window. The default is `white`.
- `foreground` (class `Foreground`)
Specifies the color to use for displaying text in the window. Setting the class name instead of the instance name is an easy way to have everything that would normally appear in the text color change color. The default is `black`.
- `cursorColor` (class `Foreground`)
Specifies the color to use for the text cursor. The default is `black`.

eightBitInput (class **EightBitInput**)

If **true**, Meta characters input from the keyboard are presented as a single character with the eighth bit turned on. If **false**, Meta characters are converted into a two-character sequence with the character itself preceded by ESC. The default is **True**.

eightBitOutput (class **EightBitOutput**)

Specifies whether or not eight-bit characters sent from the host should be accepted as is or stripped when printed. The default is **True**. (Available as of Release 5.)

font (class **Font**)

Specifies the name of the normal font. The default is **fixed**.

font1 (class **Font1**)

Specifies the name of the first alternative font. This font is toggled using the Unreadable menu item on the VT Fonts menu.

font2 (class **Font2**)

Specifies the name of the second alternative font. This font is toggled using the Tiny menu item on the VT Fonts menu.

font3 (class **Font3**)

Specifies the name of the third alternative font. This font is toggled using the Small menu item on the VT Fonts menu.

font4 (class **Font4**)

Specifies the name of the fourth alternative font. This font is toggled using the Medium menu item on the VT Fonts menu.

font5 (class **Font5**)

Specifies the name of the fifth alternative font. This font is toggled using the Large menu item on the VT Fonts menu. (Available as of Release 5.)

font6 (class **Font6**)

Specifies the name of the sixth alternative font. This font is toggled using the Huge menu item on the VT Fonts menu. (Available as of Release 5.)

geometry (class **Geometry**)

Specifies the preferred size and position of the VT102 window.

internalBorder (class **BorderWidth**)

Specifies the number of pixels between the characters and the window border. The default is 2.

jumpScroll (class **JumpScroll**)

Specifies whether or not jump scroll should be used. The default is **True**.

logFile (class **Logfile**)

Specifies the name of the file to which a terminal session is logged. The default is **XtermLog.xxxx** (where **xxxx** is the process ID of *xterm*).

- logging** (class `Logging`)
Specifies whether or not a terminal session should be logged. The default is `False`.
- logInhibit** (class `LogInhibit`)
Specifies whether or not terminal session logging should be inhibited. The default is `False`.
- loginShell** (class `LoginShell`)
Specifies whether or not the shell to be run in the window should be started as a login shell. The default is `False`.
- marginBell** (class `MarginBell`)
Specifies whether or not the bell should be rung when the user types near the right margin. The default is `False`.
- multiClickTime** (class `MultiClickTime`)
Specifies the maximum time in milliseconds between multi-click select events. The default is 250 milliseconds.
- multiScroll** (class `MultiScroll`)
Specifies whether or not scrolling should be done asynchronously. The default is `False`.
- nMarginBell** (class `Column`)
Specifies the number of characters from the right margin at which the margin bell should be rung, when enabled.
- pointerColor** (class `Foreground`)
Specifies the color of the pointer. The default is `XtDefaultForeground` color.
- pointerColorBackground** (class `Background`)
Specifies the background color of the pointer. The default is `XtDefaultBackground` color.
- pointerShape** (class `Cursor`)
Specifies the name of the shape of the pointer. The default is "xterm."
- reverseVideo** (class `ReverseVideo`)
Specifies whether or not reverse video should be simulated. The default is `False`.
- resizeGravity** (class `ResizeGravity`)
Affects the behavior when the window is resized to be taller or shorter. Acceptable values are `NorthWest` and `SouthWest`. `NorthWest` specifies that the top line of text on the screen stay fixed. If the window is made shorter, lines are dropped from the bottom; if the window is made taller, blank lines are added at the bottom. (This is compatible with the behavior in R4.) `SouthWest` (the default) specifies that the bottom line of text on the screen stay fixed. If the window is made taller, additional saved lines will be scrolled down onto the screen; if the window is made shorter, lines will be scrolled off the top of the screen, and the top saved lines will be dropped. (Available as of Release 5.)

- `reverseWrap` (class `ReverseWrap`)
Specifies whether or not reverse-wraparound should be enabled. The default is `False`.
- `saveLines` (class `SaveLines`)
Specifies the number of lines to save beyond the top of the screen when a scrollbar is turned on. The default is 64.
- `scrollBar` (class `ScrollBar`)
Specifies whether or not the scrollbar should be displayed. The default is `False`.
- `scrollTtyOutput` (class `ScrollCond`)
Specifies whether or not output to the terminal should automatically cause the scrollbar to go to the bottom of the scrolling region. The default is `True`. (In Release 5, this resource was `scrollInput`; renamed in Release 5.)
- `scrollKey` (class `ScrollCond`)
Specifies whether or not pressing a key should automatically cause the scrollbar to go to the bottom of the scrolling region. The default is `False`.
- `scrollLines` (class `ScrollLines`)
Specifies the number of lines that the `scroll-back` and `scroll-forw` actions should use as a default. The default value is 1. (See "Actions.")
- `signalInhibit` (class `SignalInhibit`)
Specifies whether or not the entries in the Main Options menu for sending signals to *xterm* should be disallowed. The default is `False`.
- `tekGeometry` (class `Geometry`)
Specifies the preferred size and position of the Tektronix window.
- `tekInhibit` (class `TekInhibit`)
Specifies whether or not Tektronix mode should be disallowed. The default is `False`.
- `tekSmall` (class `TekSmall`)
Specifies whether or not the Tektronix mode window should start in its smallest size if no explicit geometry is given. This is useful when running *xterm* on displays with small screens. The default is `False`.
- `tekStartup` (class `TekStartup`)
Specifies whether or not *xterm* should start up in Tektronix mode. The default is `False`.
- `titeInhibit` (class `TiteInhibit`)
Specifies whether or not *xterm* should remove `ti` and `te` termcap entries (used to switch between alternate screens on startup of many screen-oriented programs) from the TERMCAP string. If set, *xterm* also ignores the escape sequence to switch to the alternate screen.

`translations` (class `Translations`)

Specifies the key and button bindings for menus, selections, "programmed strings," etc. See "Actions" below.

`visualBell` (class `VisualBell`)

Specifies whether or not a visible bell (i.e., flashing) should be used instead of an audible bell when Control-G is received. The default is `False`.

The following resources are specified as part of the `tek4014` widget (class `Tek4014`):

`width` (class `Width`)

Specifies the width of the Tektronix window in pixels.

`height` (class `Height`)

Specifies the height of the Tektronix window in pixels.

`fontLarge` (class `Font`)

Specifies the large font to use in the Tektronix window. This font is toggled using the Large Characters item on the Tek Options menu.

`font2` (class `Font`)

Specifies font number 2 to use in the Tektronix window. This font is toggled using the #2 Size Characters item on the Tek Options menu.

`font3` (class `Font`)

Specifies font number 3 to use in the Tektronix window. This font is toggled using the #3 Size Characters item on the Tek Options menu.

`fontSmall` (class `Font`)

Specifies the small font to use in the Tektronix window. This font is toggled using the Small Characters item on the Tek Options menu.

`initialFont` (class `InitialFont`)

Specifies which of the four Tektronix fonts to use initially. Values are the same as for the `set-tek-text` action. The default is `large`. (Available as of Release 5.)

`ginTerminator` (class `GinTerminator`)

Specifies what character(s) should follow a GIN report or status report. Acceptable values are `none`, which sends no terminating characters, `CRonly`, which sends CR, and `CR&EOT`, which sends both CR and EOT. The default is `none`. (Available as of Release 5.)

The resources that can be specified for the various menus are described in the documentation for the Athena `SimpleMenu` widget. The name and classes of the entries in each of the menus are listed below.

The `mainMenu` (title Main Options) has the following entries:

`securekbd` (class `SmeBSB`)

Invokes the `secure()` action.

`allowsends` (class `SmeBSB`)
Invokes the `allow-send-events(toggle)` action.

`logging` (class `SmeBSB`)
Invokes the `set-logging(toggle)` action.

`redraw` (class `SmeBSB`)
Invokes the `redraw()` action.

`line1` (class `SmeLine`)
A separator.

`suspend` (class `SmeBSB`)
Invokes the `send-signal(tstp)` action on systems that support job control.

`continue` (class `SmeBSB`)
Invokes the `send-signal(cont)` action on systems that support job control.

`interrupt` (class `SmeBSB`)
Invokes the `send-signal(int)` action.

`hangup` (class `SmeBSB`)
Invokes the `send-signal(hup)` action.

`terminate` (class `SmeBSB`)
Invokes the `send-signal(term)` action.

`kill` (class `SmeBSB`)
Invokes the `send-signal(kill)` action.

`line2` (class `SmeLine`)
A separator.

`quit` (class `SmeBSB`)
Invokes the `quit()` action.

The `vtMenu` (title `VT Options`) has the following entries:

`scrollbar` (class `SmeBSB`)
Invokes the `set-scrollbar(toggle)` action.

`jumpscroll` (class `SmeBSB`)
Invokes the `set-jumpscroll(toggle)` action.

`reversevideo` (class `SmeBSB`)
Invokes the `set-reverse-video(toggle)` action.

`autowrap` (class `SmeBSB`)
Invokes the `set-autowrap(toggle)` action.

`reversewrap` (class `SmeBSB`)
Invokes the `set-reversewrap(toggle)` action.

`autolinefeed` (class `SmeBSB`)
 Invokes the `set-autolinefeed(toggle)` action.

`appcursor` (class `SmeBSB`)
 Invokes the `set-appcursor(toggle)` action.

`appkeypad` (class `SmeBSB`)
 Invokes the `set-appkeypad(toggle)` action.

`scrollkey` (class `SmeBSB`)
 Invokes the `set-scroll-on-key(toggle)` action.

`scrollttyoutput` (class `SmeBSB`)
 Invokes the `set-scroll-on-tty-output(toggle)` action.

`allow132` (class `SmeBSB`)
 Invokes the `set-allow132(toggle)` action.

`cursesemul` (class `SmeBSB`)
 Invokes the `set-cursesemul(toggle)` action.

`visualbell` (class `SmeBSB`)
 Invokes the `set-visualbell(toggle)` action.

`marginbell` (class `SmeBSB`)
 Invokes the `set-marginbell(toggle)` action.

`altscreen` (class `SmeBSB`)
 This entry is currently disabled.

`line1` (class `SmeLine`)
 A separator.

`softreset` (class `SmeBSB`)
 Invokes the `soft-reset()` action.

`hardreset` (class `SmeBSB`)
 Invokes the `hard-reset()` action.

`clearsavedlines` (class `SmeBSB`)
 Invokes the `clear-saved-lines()` action. (Available as of Release 5.)

`line2` (class `SmeLine`)
 A separator.

`tekshow` (class `SmeBSB`)
 Invokes the `set-visibility(tek,toggle)` action.

`tekmode` (class `SmeBSB`)
 Invokes the `set-terminal-type(tek)` action.

`vthide` (class `SmeBSB`)
 Invokes the `set-visibility(vt,off)` action.

The tekMenu (title Tek Options) has the following entries:

tektextlarge (class SmeBSB)

Invokes the set-tek-text (1) action.

tektext2 (class SmeBSB)

Invokes the set-tek-text (2) action.

tektext3 (class SmeBSB)

Invokes the set-tek-text (3) action.

tektextsmall (class SmeBSB)

Invokes the set-tek-text (s) action.

line1 (class SmeLine)

A separator.

tekpage (class SmeBSB)

Invokes the tek-page () action.

tekreset (class SmeBSB)

Invokes the tek-reset () action.

tekcopy (class SmeBSB)

Invokes the tek-copy () action.

line2 (class SmeLine)

A separator.

vtshow (class SmeBSB)

Invokes the set-visibility (vt, toggle) action.

vtmode (class SmeBSB)

Invokes the set-terminal-type (vt) action.

tekhide (class SmeBSB)

Invokes the set-visibility (tek, toggle) action.

The fontMenu (title VT Fonts) has the following entries:

fontdefault (class SmeBSB)

Invokes the set-vt-font (d) action.

font1 (class SmeBSB)

Invokes the set-vt-font (1) action.

font2 (class SmeBSB)

Invokes the set-vt-font (2) action.

font3 (class SmeBSB)

Invokes the set-vt-font (3) action.

font4 (class SmeBSB)

Invokes the set-vt-font (4) action.

- font5 (class SmeBSB)
 Invokes the set-vt-font (5) action. (Available as of Release 5.)
- font6 (class SmeBSB)
 Invokes the set-vt-font (6) action. (Available as of Release 5.)
- fontescape (class SmeBSB)
 Invokes the set-vt-font (e) action.
- fontsel (class SmeBSB)
 Invokes the set-vt-font (s) action.

The following resources are useful when specified for the Athena Scrollbar widget (scroll-Bar, class ScrollBar):

- thickness (class Thickness)
 Specifies the width in pixels of the scrollbar.
- background (class Background)
 Specifies the color to use for the background of the scrollbar.
- foreground (class Foreground)
 Specifies the color to use for the foreground of the scrollbar. The “thumb” of the scrollbar is a simple checkerboard pattern alternating pixels for foreground and background color.

Emulations

The VT102 emulation is fairly complete, but does not support the blinking character attribute nor the double-wide and double-size character sets. *termcap* entries that work with *xterm* include “xterm,” “vt102,” “vt100,” and “ansi.” *xterm* automatically searches the *termcap* file in this order for these entries and then sets the TERM and the TERMCAP environment variables. Note that the “xterm” *termcap* entry distributed with X is not automatically installed. You must add it to */etc/termcap* yourself.

Many of the special *xterm* features (like logging) may be modified under program control through a set of escape sequences different from the standard VT102 escape sequences. (See Appendix E, *xterm Control Sequences*, in this guide.)

The Tektronix 4014 emulation is also fairly good. Four different font sizes and five different line types are supported. The Tektronix text and graphics commands are recorded internally by *xterm* and may be written to a file by sending the COPY escape sequence (or through the Tektronix menu; see below). The name of the file will be “COPYyy-MM-dd.hh:mm:ss”, where yy, MM, dd, hh, mm, and ss are the year, month, day, hour, minute, and second when the COPY was performed (the file is created in the directory in which *xterm* is started, or the home directory for a login *xterm*).

Pointer Usage

Once the VT102 window is created, *xterm* allows you to select text and copy it within the same or other windows.

The selection functions are invoked when the pointer buttons are used with no modifiers, and when they are used with the Shift key. The assignment of the functions described below to keys and buttons may be changed through the resource database; see “Actions” below.

Pointer button 1 (usually the left) is used to save text into the cut buffer. Move the cursor to the beginning of the text, and then hold the button down while moving the cursor to the end of the region and release the button. The selected text is highlighted and is saved in the global cut buffer and made the PRIMARY selection when the button is released. Double-clicking selects by words. Triple-clicking selects by lines. Quadruple-clicking goes back to characters, etc. Multiple-click is determined by the time from button up to button down, so you can change the selection unit in the middle of a selection. If the key/button bindings specify that an X selection is to be made, *xterm* will leave the selected text highlighted for as long as it is the selection owner.

Pointer button 2 (usually the middle) “types” (pastes) the text from the PRIMARY selection, if any, otherwise from the cut buffer, inserting it as keyboard input.

Pointer button 3 (usually the right) extends the current selection. (You can swap “right” and “left” everywhere in the rest of this paragraph.) If pressed while closer to the right edge of the selection than the left, it extends/contracts the right edge of the selection. If you contract the selection past the left edge of the selection, *xterm* assumes you really meant the left edge, restores the original selection, then extends/contracts the left edge of the selection. Extension starts in the selection unit mode in which the last selection or extension was performed; you can multiple-click to cycle through them.

By cutting and pasting pieces of text without trailing new lines, you can take text from several places in different windows and form a command to the shell, for example, or take output from a program and insert it into your favorite editor. Since the cut buffer is globally shared among different applications, you should regard it as a “file” whose contents you know. The terminal emulator and other text programs should be treating it as if it were a text file, i.e., the text is delimited by new lines.

The scroll region displays the position and amount of text currently showing in the window (highlighted) relative to the amount of text actually saved. As more text is saved (up to the maximum), the size of the highlighted area decreases.

Clicking button 1 in the scroll region moves the adjacent line to the top of the display window.

Clicking button 3 moves the top line of the display window down to the pointer position.

Clicking button 2 moves the display to a position in the saved text that corresponds to the pointer’s position in the scrollbar.

Unlike the VT102 window, the Tektronix window does not allow the copying of text. It does allow Tektronix GIN mode, and in this mode the cursor will change from an arrow to a cross.

Pressing any key will send that key and the current coordinate of the cross cursor. Pressing button 1, 2, or 3 will return the letters “l,” “m,” and “r,” respectively. If the Shift key is pressed when a pointer button is pressed, the corresponding uppercase letter is sent. To distinguish a pointer button from a key, the high bit of the character is set (but this bit is normally stripped unless the terminal mode is RAW; see *tty(4)* for details).

Menus

The Release 5 version of *xterm* has four different menus, titled Main Options, VT Options, Tek Options, and VT Fonts. Each menu pops up under the correct combination of key and button presses. Most menus are divided into two sections, separated by a horizontal line. The top portion contains various modes that can be specified. A check mark appears next to a mode that is currently active. Selecting one of these modes toggles its state. The bottom portion contains command entries; selecting one of these performs the indicated function. The menus are described in detail in the following sections.

Main Options Menu

The Main Options menu is displayed when the Control key and pointer button 1 are simultaneously pressed in an *xterm* window. The modes section contains items that apply to both the VT102 and Tektronix windows. The modes can also be set by command-line options when invoking *xterm*, or by entries in a resource startup file like *.Xresources* (see Chapter 11, *Setting Resources*). The menu selections enable you to change your mind once *xterm* is running.

All of the commands on this menu (except for Redraw Window) send a signal that is intended to affect the *xterm* process (Send INT Signal, Send TERM Signal, etc.). Given that your operating system may recognize only certain signals, every menu item may not produce the intended function.

Four of these commands (Send HUP Signal, Send TERM Signal, Send KILL Signal, and Quit) send signals that are intended to terminate the *xterm* window. In most cases, you can probably end an *xterm* process simply by typing some sequence (such as Control-D or *exit*) in the window. Of course, the menu options may be helpful if the more conventional ways of killing the window fail. Refer to the section on *xkill* in Chapter 8, *Other Clients*, for a discussion of the hazards of killing a client and a summary of alternatives.

Main Options Menu Mode Toggles (On/Off)

Secure Keyboard	Ensures that all keyboard input is directed <i>only</i> to <i>xterm</i> . Used when typing in passwords or other sensitive data in an unsecure environment. (See “Security” later in this reference page.)
Allow SendEvents	Causes synthetic key and button events (generated using the X protocol <i>SendEvent</i> request) to be interpreted. Note that allowing such events creates a very large security hole.
Log to File	Logs <i>xterm</i> input/output into a file in your home directory called <i>XtermLog.xxxxx</i> where <i>xxxxx</i> represents the process ID number of the <i>xterm</i> process. Logging allows you to keep track of the sequence of data and, therefore, is particularly helpful while debugging code.

To display the data contained in the log file, at the *xterm* window prompt type:

```
more XtermLog.xxxxx
```

The output file is sent to your *xterm* window.

Be sure to turn Log to File off before displaying the log file in the *xterm* window. When Log to File is on, anything in the window is appended to the end of the log file. If you display the log file while logging is on, you will get into a continuous loop, much as if you typed `cat * > file`.

To find out the exact name of the log file, list the contents of your home directory, looking for a log file with an appropriate time and date. Note that if you turn logging on in multiple *xterm* windows, there will be multiple log files.

Main Options Menu Commands

- | | |
|------------------|---|
| Redraw Window | Redraws the contents of the window. (You can redraw the entire screen using the <i>xrefresh</i> client. If you are running the <i>mwm</i> window manager, you can also do this using the Refresh item on <i>mwm</i> 's Root Menu.) |
| Send STOP Signal | Suspends a process (sends the SIGTSTP signal to the process group of the process running under <i>xterm</i> , usually the shell). If your system supports job control, you may also be able to suspend the process by typing Control-Z. If your system does not support job control, this menu item won't work either. |
| Send CONT Signal | Continues a process that has been suspended (technically speaking, this menu item sends the SIGCONT signal to the process group of the process running under <i>xterm</i> , usually the shell). The Send CONT Signals item is especially useful on systems with job control if you accidentally type Control-Z and suspend a process. |
| Send INT Signal | Interrupts a process (sends the SIGINT signal to the process group of the process running under <i>xterm</i> , usually the shell). |
| Send HUP Signal | Hangs up the process (sends the SIGHUP signal to the process group of the process running under <i>xterm</i> , usually the shell). This usually ends up killing the <i>xterm</i> process, and the window disappears from the screen. |
| Send TERM Signal | Terminates the process (sends the SIGTERM signal to the process group of the process running under <i>xterm</i> , usually the shell). This usually ends up killing the <i>xterm</i> process, and the window disappears from the screen. |

- Send KILL Signal Kills the process (sends the SIGKILL signal to the process group of the process running under *xterm*, usually the shell). This ends up killing the *xterm* process, and the window disappears from the screen.
- Quit Like Send HUP Signal, Quit sends the SIGHUP signal to the process group of the process running under *xterm*, usually the shell. This usually ends up killing the *xterm* process, and the window disappears from the screen.
- Quit is separated from the earlier commands by a horizontal line, so it's easier to point at. Sending a SIGHUP signal with Quit is also slightly more gentle to the system than using Send KILL Signal.

See *signal(3C)* in the *UNIX Programmer's Manual* for more information on what each signal does.

VT Options Menu

The VT Options menu sets various characteristics (or modes) of the VT102 emulation window and is displayed when the Control key and pointer button 2 are pressed in the VT102 window.

In the command section of this menu, the soft reset entry will reset scroll regions. This can be convenient when some program has left the scroll regions set incorrectly (often a problem when using VMS or TOPS-20). The full reset entry will clear the screen, reset tabs to every eight columns, and reset the terminal modes (such as wrap and smooth scroll) to their initial states just after *xterm* has finish processing the command-line options.

VT Options Menu Mode Toggles (On/Off)

Most of these modes can also be set by command-line options when invoking *xterm* or by entries in a resource startup file like *.Xresources*. (See "Options" and "Resources" in this reference page. See Chapter 11, *Setting Resources*, for information on resource files and syntax.) The menu selections enable you to change your mind about the characteristics of an *xterm* after the window is running.

- Enable Scrollbar Causes a scrollbar to appear on the left-hand side of the *xterm* window. Off by default.
- Enable Jump Scroll Causes the window to move text several lines at a time rather than line by line. On by default.
- Enable Reverse Video Reverses the foreground and background colors. Off by default.
- Enable Auto Wraparound Wraps the text or data to the next line automatically when the cursor reaches the window border on input. On by default.
- Enable Reverse Wraparound Allows the cursor to wrap around from the leftmost column to the rightmost column of the previous line. Allows you to backspace to

the previous line and overstrike data or erase data with the space bar. Off by default.

Enable Auto Linefeed Generates a linefeed automatically. This is useful if you are using a program that generates a carriage return without dropping down a line on your screen. Off by default. (This option is usually not needed on UNIX systems.)

Enable Application Cursor Keys Generates ANSI escape sequences rather than standard cursor movement when you use the arrow keys. This option may be useful when working with certain applications. Off by default.

The following table lists the ANSI characters generated by application cursors.

Cursor Key (Arrow)	Reset (Cursor)	Set (Application)
Up	ESC [A	ESC O A
Down	ESC [B	ESC O B
Right	ESC [C	ESC O C
Left	ESC [D	ESC O D

Enable Application Keypad Generates a control function rather than a numeric character when you use the numeric keypad. Off by default.

Scroll to Bottom on Key Press Indicates that pressing a key while using the scrollbar causes the cursor to be repositioned at the bottom of the scroll region. For example, if you have scrolled up the window to see past history, as soon as you begin typing your next command the cursor jumps to the bottom of the screen. Off by default.

Scroll to Bottom on Tty Output Indicates that receiving output to the window (or pressing a key, if `stty echo` has been specified) while using the scrollbar causes the cursor to be repositioned at the bottom of the scroll region. On by default. This mode can be toggled off, but is generally desirable to have.

Allow 80/132 Column Switching

Allows *xterm* to recognize the DECCOLM escape sequence, which switches the terminal between 80- and 132-column mode. The DECCOLM escape sequence can be included in a program (such as a spreadsheet) to allow the program to display in 132-column format. See Appendix E, *xterm Control Sequences*, for more information. Off by default.

Enable Curses Emulation

Enables the *curses* fix. Several programs that use the *curses* cursor motion package have some difficulties with VT102-compatible terminals. The bug occurs when you run the *more* program on a file containing a line that is exactly the width of the window and that is followed by a line beginning with a tab. The leading tabs may disappear. This mode causes the tabs to be displayed correctly. Off by default.

Enable Visual Bell

Causes your terminal window to flash whenever an event occurs that would ordinarily cause your terminal bell to ring.

Enable Margin Bell

Turns on the margin bell. Off by default.

Tek Window Showing

Shows the current contents of the Tektronix window; you cannot input to that window until you choose Switch to Tek Mode. Off by default.

Show Alternate Screen

Informs you that you are looking at the alternate screen. You cannot select this mode from the menu. If a check mark appears beside this mode, you are viewing the alternate screen. Off by default.

VT Options Menu Commands

These commands can be invoked only from the menu; there are no alternative ways to perform the same functions.

Do Soft Reset

Resets the terminal scroll region from partial scroll (a portion of the window) to full scroll (the entire window). Use this command when a program has left the scroll region set incorrectly.

Do Full Reset

Clears the window, resets tabs to every eight columns, and resets the terminal modes such as auto wraparound and jump scroll to their initial states.

Reset and Clear Saved Lines

Does full reset (see above) and also clears the history of lines saved off the top of the screen. (Available as of Release 5.)

Show Tek Window

Shows the current contents of the Tektronix window; you cannot input to that window until you choose Switch to Tek Mode. Off by default.

Switch to Tek Mode

Brings up a Tektronix window. You can input to this window.

Window Terminal Emulator**xterm** (continued)

Hide VT Window Removes the VT window but does not destroy it. It can be brought back by choosing Select VT Mode from the Tek Options menu.

Tek Options Menu

The Tek Options menu (formerly Tektronix) sets various modes in the Tektronix emulation, and is displayed when the Control key and pointer button 2 are pressed in the Tektronix window. The current font size is checked in the modes section of the menu. The PAGE entry in the command section clears the Tektronix window.

Tek Options Menu Mode Toggles (On/Off)

These modes can be set only from the Tek Options menu.

Large Characters Selecting one of these four options sets the point size of text displayed in the Tektronix window. The four options are mutually exclusive.
 #2 Size Characters
 #3 Size Characters
 Small Characters

Tek Options Menu Commands

PAGE Clears the Tektronix window.
 RESET Closes down the Tektronix window.
 COPY Writes a file of the Tektronix text and graphics commands.
 Show VT Window Shows the current contents of the VT102 window; you cannot input to that window until you choose Switch to VT Mode.
 Switch to VT Mode Makes the associated VT102 window active for input.
 Hide Tek Window Removes the Tektronix window but does not destroy it. It can be brought back by choosing Switch to Tek Mode from the VT Options menu.

VT Fonts Menu

The VT Fonts menu enables you to change the VT102 display font dynamically. The menu is displayed when the Control key and pointer button 3 are pressed in the VT102 window. All items on the menu toggle different display fonts. The items are mutually exclusive. A checkmark appears on the menu next to the current font.

In addition to the default font and a number of alternatives that are set with resources, the menu offers: the Escape Sequence menu item, which toggles the font last specified by the Set Font escape sequence; and the Selection menu item, which tries to use the current text selection as a font name (if the PRIMARY text selection is owned). Chapter 6, *Font Specification*, explains how to use these menu items.

- Default Selecting one of these seven options sets the point size of text displayed in the Unreadable VT102 window. The Default font is the font specified when the *xterm* was run.
- Tiny
- Small
- Medium
- Large
- Huge
- Escape Sequence Allows you to select a font previously toggled using an escape sequence. See Chapter 6, *Font Specification*, for the escape sequence to use.
- Selection Allows you to toggle a font whose name you've previously selected with the pointer or using the select button of the *xfonstsel* client. See Chapter 6, *Font Specification*, for more information.

Security

X environments differ in their security consciousness. MIT servers, run under *xdm*, are capable of using a "magic cookie" authorization scheme that can provide a reasonable level of security for many people. If your server is using only a host-based mechanism to control access to the server (see *xhost*), then if you enable access for a host and other users are also permitted to run clients on that same host, there is every possibility that someone can run an application that will use the basic services of the X protocol to snoop on your activities, potentially capturing a transcript of everything you type at the keyboard. This is of particular concern when you want to type in a password or other sensitive data. The best solution to this problem is to use a better authorization mechanism than host-based control, but a simple mechanism exists for protecting keyboard input in *xterm*.

The Main Options menu (see "Menus" above) contains a Secure Keyboard entry which, when enabled, ensures that all keyboard input is directed *only* to *xterm* (using the GrabKeyboard protocol request). When an application prompts you for a password (or other sensitive data), you can enable Secure Keyboard using the menu, type in the data, and then disable Secure Keyboard using the menu again. Only one X client at a time can secure the keyboard, so when you attempt to enable Secure Keyboard it may fail. In this case, the bell will sound. If the Secure Keyboard succeeds, the foreground and background colors will be exchanged (as if you selected the Enable Reverse Video entry in the VT Options menu); they will be exchanged again when you exit secure mode. If the colors do *not* switch, then you should be *very* suspicious that you are being spoofed. If the application you are running displays a prompt before asking for the password, it is safest to enter secure mode *before* the prompt gets displayed, and to make sure that the prompt gets displayed correctly (in the new colors), to minimize the probability of spoofing. You can also bring up the menu again and make sure that a check mark appears next to the entry.

Secure Keyboard mode will be disabled automatically if your xterm window becomes iconified (or otherwise unmapped), or if you start up a reparenting window manager (that places a titlebar or other decoration around the window) while in Secure Keyboard mode. (This is a feature of the X protocol not easily overcome.) When this happens, the foreground and background colors will be switched back and the bell will sound in warning.

Character Classes

Clicking the middle mouse button twice in rapid succession will cause all characters of the same class (e.g., letters, white space, punctuation) to be selected. Since different people have different preferences for what should be selected (for example, should filenames be selected as a whole or only the separate subnames), the default mapping can be overridden through the use of the `charClass` (class `CharClass`) resource.

This resource is simply a list of *range:value* pairs, where the range is either a single number or *low-high* in the range of 0 to 127, corresponding to the ASCII code for the character or characters to be set. The *value* is arbitrary, although the default table uses the character number of the first character occurring in the set.

The default table is:

```
static int charClass[128] = {
/* NUL  SOH  STX  ETX  EOT  ENQ  ACK  BEL */
   32,  1,  1,  1,  1,  1,  1,  1,
/* BS   HT   NL   VT   NP   CR   SO   SI */
   1,  32,  1,  1,  1,  1,  1,  1,
/* DLE  DC1  DC2  DC3  DC4  NAK  SYN  ETB */
   1,  1,  1,  1,  1,  1,  1,  1,
/* CAN  EM   SUB  ESC  FS   GS   RS   US */
   1,  1,  1,  1,  1,  1,  1,  1,
/* SP   !    →    #    $    %    &    ' */
   32,  33,  34,  35,  36,  37,  38,  39,
/* (    )    *    +    ,    -    .    / */
   40,  41,  42,  43,  44,  45,  46,  47,
/* 0    1    2    3    4    5    6    7 */
   48,  48,  48,  48,  48,  48,  48,  48,
/* 8    9    :    ;    <    =    >    ? */
   48,  48,  58,  59,  60,  61,  62,  63,
/* @    A    B    C    D    E    F    G */
   64,  48,  48,  48,  48,  48,  48,  48,
/* H    I    J    K    L    M    N    O */
   48,  48,  48,  48,  48,  48,  48,  48,
/* P    Q    R    S    T    U    V    W */
   48,  48,  48,  48,  48,  48,  48,  48,
/* X    Y    Z    [    \    ]    ^    _ */
   48,  48,  48,  91,  92,  93,  94,  48,
/* `    a    b    c    d    e    f    g */
   96,  48,  48,  48,  48,  48,  48,  48,
```

```

/*  h   i   j   k   l   m   n   o */
   48, 48, 48, 48, 48, 48, 48, 48,
/*  p   q   r   s   t   u   v   w */
   48, 48, 48, 48, 48, 48, 48, 48,
/*  x   y   z   {   |   }   ~ DEL */
   48, 48, 48, 123, 124, 125, 126, 1};

```

For example, the string “33:48,37:48,45-47:48,64:48” indicates that the exclamation mark, percent sign, dash, period, slash, and ampersand characters should be treated the same way as characters and numbers. This is very useful for cutting and pasting electronic mailing addresses and UNIX filenames.

Actions

It is possible to rebind keys (or sequences of keys) to arbitrary strings for input, by changing the translations for the vt100 or tek4014 widgets. Changing the translations for events other than key and button events is not expected, and will cause unpredictable behavior. The following actions are provided for use with the vt100 or tek4014 translations resource:

`bell([percent])`

Rings the keyboard bell at the specified percentage above or below the base volume.

`ignore()`

Ignores the event but checks for special pointer position escape sequences. This is useful for trapping events that might otherwise interfere with translations you might want to set.

`insert()`

Inserts the character or string associated with the key that was pressed.

`insert-seven-bit()`

A synonym for `insert()`.

`insert-eight-bit()`

Inserts the eight-bit (Meta) version of the character or string associated with the key that was pressed. (The fallback translation associated with this action is `Meta<KeyPress>`. That is, pressing Meta in conjunction with any key will get the 8-bit equivalent.) The exact action depends on the value of the `eightBitInput` resource.

`insert-selection(sourcename [, . . .])`

Inserts the string found in the selection or cut buffer indicated by *sourcename*. Sources are checked in the order given (case is significant) until one is found. Commonly-used selections include: PRIMARY, SECONDARY, and CLIPBOARD. Cut buffers are typically named CUT_BUFFER0 through CUT_BUFFER7.

`keymap(name)`

Dynamically defines a new translation table whose resource name is *name* with the suffix `Keymap` (case is significant). The keymap name `None` restores the original

translation table. This is useful for loading translations that will be used with a particular application running in an *xterm* window. In the following example, *keymap* is used to define a set of special keys for entering commonly-typed words when running the *dbx* application:

```
*VT100.Translations: #override <Key>F13: keymap(dbx)
*VT100.dbxKeymap.translations: \
```

```
<Key> F14:  keymap(None) \n\
<Key> F17:  string("next") string(0x0d) \n\
<Key> F18:  string("step") string(0x0d) \n\
<Key> F19:  string("continue") string(0x0d) \n\
<Key> F20:  string("print ") insert-selection(PRIMARY,CUT_BUFFER0)
```

When the user presses key F13, the *dbx* keymaps go into effect. Keys F15-F20 then print common *dbx* commands. F14 disables the translations on the “*dbx*” keys.

popup-menu (*menuname*)

Displays the specified popup menu. Valid names (case is significant) include: *mainMenu*, *vtMenu*, *fontMenu*, and *tekMenu*.

secure()

Toggles the secure keyboard mode described in the “Security” section, and is invoked from the Secure Keyboard entry in *mainMenu*.

select-start()

Begins text selection at the current pointer location. See the section on “Pointer Usage” for information on making selections.

select-extend()

Tracks the pointer and extends the selection. It should be bound only to motion events.

select-end(*destname* [, . . .])

Puts the currently selected text into all of the selections or cut buffers specified by *destname*.

select-cursor-start()

Similar to *select-start*, except that it begins the selection at the current text cursor position.

select-cursor-end(*destname* [, . . .])

Similar to *select-end*, except that it should be used with *select-cursor-start*.

set-vt-font (*d*/1/2/3/4/5/6/*e*/*s* [, *normalfont* [, *boldfont*]])

Sets the font or fonts currently being used in the VT102 window. The first argument is a single character that specifies the font to be used: *d* or *D* indicates the default font (the font initially used when *xterm* was started); 1 through 6 indicate the fonts

specified by the `font1` through `font6` resources; `e` or `E` indicates the normal and bold fonts that may be set through escape codes (or specified as the second and third action arguments, respectively); and `s` or `S` indicates the font selection (as made by programs such as *xfontsel*) indicated by the second action argument.

`start-extend()`

Similar to `select-start` except that the selection is extended to the current pointer location.

`start-cursor-extend()`

Similar to `select-extend` except that the selection is extended to the current text cursor position.

`string(string)`

Inserts the specified text string as if it had been typed. Quotation is necessary if the string contains whitespace or non-alphanumeric characters. If the string argument begins with the characters "0x", it is interpreted as a hex character constant.

`scroll-back(count [, units])`

Scrolls the text window backward so that text that had previously scrolled off the top of the screen is now visible. The `count` argument indicates the number of `units` (which may be `page`, `halfpage`, `pixel`, or `line`) by which to scroll.

`scroll-forw(count [, units])`

Scrolls is similar to `scroll-back` except that it scrolls in the other direction.

`allow-send-events(on/off/toggle)`

Sets or toggles the `allowSendEvents` resource and is also invoked by the `allowsends` entry in `mainMenu`.

`set-logging(on/off/toggle)`

Toggles the `logging` resource and is also invoked by the `logging` entry in `mainMenu`.

`redraw()`

Redraws the window and is also invoked by the `redraw` entry in `mainMenu`.

`send-signal(signame)`

Sends the signal named by `signame` (which may also be a number) to the `xterm` subprocess (the shell or program specified with the `-e` command-line option) and is also invoked by the `suspend`, `continue`, `interrupt`, `hangup`, `terminate`, and `kill` entries in `mainMenu`. Allowable signal names are (case is not significant): `tstp` (if supported by the operating system), `suspend` (same as `tstp`), `cont` (if supported by the operating system), `int`, `hup`, `term`, `quit`, `alarm`, `alarm` (same as `alarm`) and `kill`.

`quit()`

Sends a `SIGHUP` to the subprogram and exits. It is also invoked by the `quit` entry in `mainMenu`.

- `set-scrollbar (on/off/toggle)`
Toggles the scrollbar resource and is also invoked by the scrollbar entry in vtMenu.
- `set-jumpscroll (on/off/toggle)`
Toggles the jumpscroll resource and is also invoked by the jumpscroll entry in vtMenu.
- `set-reverse-video (on/off/toggle)`
Toggles the reverseVideo resource and is also invoked by the reversevideo entry in vtMenu.
- `set-autowrap (on/off/toggle)`
Toggles automatic wrapping of long lines and is also invoked by the autowrap entry in vtMenu.
- `set-reversewrap (on/off/toggle)`
Toggles the reverseWrap resource and is also invoked by the reversewrap entry in vtMenu.
- `set-autolinefeed (on/off/toggle)`
Toggles automatic insertion of linefeeds and is also invoked by the autolinefeed entry in vtMenu.
- `set-appcursor (on/off/toggle)`
Toggles the application cursor key mode and is also invoked by the appcursor entry in vtMenu.
- `set-appkeypad (on/off/toggle)`
Toggles the application keypad mode and is also invoked by the appkeypad entry in vtMenu.
- `set-scroll-on-key (on/off/toggle)`
Toggles the scrollKey resource and is also invoked from the scrollkey entry in vtMenu.
- `set-scroll-on-tty-output (on/off/toggle)`
Toggles the scrollTtyOutput resource and is also invoked from the scroll-ttyoutput entry in vtMenu.
- `set-allow132 (on/off/toggle)`
Toggles the c132 resource and is also invoked from the allow132 entry in vtMenu.
- `set-cursesemul (on/off/toggle)`
Toggles the curses resource and is also invoked from the cursesemul entry in vtMenu.

`set-visual-bell (on/off/toggle)`

Toggles the `visualBell` resource and is also invoked by the `visualbell` entry in `vtMenu`.

`set-marginbell (on/off/toggle)`

Toggles the `marginBell` resource and is also invoked from the `marginbell` entry in `vtMenu`.

`set-altscreen (on/off/toggle)`

Toggles between the alternate and current screens.

`soft-reset ()`

Resets the scrolling region and is also invoked from the `softreset` entry in `vtMenu`.

`hard-reset ()`

Resets the scrolling region, tabs, window size, and cursor keys and clears the screen. It is also invoked from the `hardreset` entry in `vtMenu`.

`clear-saved-lines ()`

Does `hard-reset` (see above) and also clears the history of lines saved off the top of the screen. (Available as of Release 5.) This action is also invoked from the `clearsavedlines` entry in the `vtMenu`.

`set-terminal-type (type)`

Directs output to either the `vt` or `tek` windows, according to the `type` string. It is also invoked by the `tekmode` entry in `vtMenu` and the `vtmode` entry in `tekMenu`.

`set-visibility (vt/tek, on/off/toggle)`

Controls whether or not the `vt` or `tek` windows are visible. It is also invoked from the `tekshow` and `vthide` entries in `vtMenu` and the `vtshow` and `tekhide` entries in `tekMenu`.

`set-tek-text (large/2/3/small)`

Sets font used in the Tektronix window to the value of the resources `tektextlarge`, `tektext2`, `tektext3`, and `tektextsmall` according to the argument. It is also invoked by the entries of the same names as the resources in `tekMenu`.

`tek-page ()`

Clears the Tektronix window and is also invoked by the `tekpage` entry in `tekMenu`.

`tek-reset ()`

Resets the Tektronix window and is also invoked by the `tekreset` entry in `tekMenu`.

`tek-copy ()`

Copies the escape codes used to generate the current window contents to a file in the current directory beginning with the name `COPY`. It is also invoked from the `tekcopy` entry in `tekMenu`.

visual-bell()

Flashes the window quickly. (Available as of Release 5.)

The Tektronix window also has the following action:

gin-press(l/L/m/M/r/R)

Sends the indicated graphics input code.

The default bindings in the VT102 window are:

Shift	<KeyPress>	Prior:	scroll-back(1,halpage) \n\
Shift	<KeyPress>	Next:	scroll-forw(1,halpage) \n\
Shift	<KeyPress>	Select:	select-cursor-start() \n\
			select-cursor-end(PRIMARY,CUT_BUFFER0) \n\
Shift	<KeyPress>	Insert:	insert-selection(PRIMARY,CUT_BUFFER0) \n\
	~Meta	<KeyPress>:	insert-seven-bit() \n\
	Meta	<KeyPress>:	insert-eight-bit() \n\
	!Ctrl	<Btn1Down>:	popup-menu(mainMenu) \n\
!Lock	Ctrl	<Btn1Down>:	popup-menu(mainMenu) \n\
	~Meta	<Btn1Down>:	select-start() \n\
	~Meta	<Btn1Motion>:	select-extend() \n\
	!Ctrl	<Btn2Down>:	popup-menu(vtMenu) \n\
!Lock	Ctrl	<Btn2Down>:	popup-menu(vtMenu) \n\
~Ctrl	~Meta	<Btn2Down>:	ignore() \n\
~Ctrl	~Meta	<Btn2Up>:	insert-selection(PRIMARY,CUT_BUFFER0) \n\
	!Ctrl	<Btn3Down>:	popup-menu(fontMenu) \n\
!Lock	Ctrl	<Btn3Down>:	popup-menu(fontMenu) \n\
~Ctrl	~Meta	<Btn3Down>:	start-extend() \n\
	~Meta	<Btn3Motion>:	select-extend() \n\
		<BtnUp>:	select-end(PRIMARY,CUT_BUFFER0) \n\
		<BtnDown>:	bell(0)

The default bindings in the Tektronix window are:

	~Meta	<KeyPress>:	insert-seven-bit() \n\
	Meta	<KeyPress>:	insert-eight-bit() \n\
	!Ctrl	<Btn1Down>:	popup-menu(mainMenu) \n\
!Lock	Ctrl	<Btn1Down>:	popup-menu(mainMenu) \n\
	!Ctrl	<Btn2Down>:	popup-menu(tekMenu) \n\
!Lock	Ctrl	<Btn2Down>:	popup-menu(tekMenu) \n\
Shift	~Meta	<Btn1Down>:	gin-press(L) \n\
	~Meta	<Btn1Down>:	gin-press(l) \n\
Shift	~Meta	<Btn2Down>:	gin-press(M) \n\
	~Meta	<Btn2Down>:	gin-press(m) \n\
Shift	~Meta	<Btn3Down>:	gin-press(R) \n\
	~Meta	<Btn3Down>:	gin-press(r)

Environment

xterm sets the environment variables TERM and TERMCAP properly for the size window you have created. It also uses and sets the environment variable DISPLAY to specify which bitmap display terminal to use. The environment variable WINDOWID is set to the X window ID number of the *xterm* window.

Bugs

The class name is XTerm instead of Xterm.

Large pastes do not work on some systems. This is not a bug in *xterm*; it is a bug in the pseudo-terminal driver of those systems. *xterm* feeds large pastes to the pseudo-terminal only as fast as the *pty* will accept data, but some *pty* drivers do not return enough information to know if the write has succeeded.

Many of the options are not resettable after *xterm* starts (i.e., the menus allow you to change only some of *xterm*'s features dynamically).

The Tek widget does not support key/button re-binding.

Only fixed-width, character cell fonts are supported.

This program still needs to be rewritten. It should be split into very modular sections, with the various emulators being completely separate widgets that don't know about each other. Ideally, you'd like to be able to pick and choose emulator widgets and stick them into a single control widget.

There needs to be a dialog box to allow entry of the log file name and the COPY filename.

See Also

X, resize, pty(4), tty(4); Chapter 5, *The xterm Terminal Emulator*; Chapter 6, *Font Specification*; Appendix E, *xterm Control Sequences*.

Authors

Far too many people, including:

Loretta Guarino Reid (DEC-UEG-WSL), Joel McCormack (DEC-UEG-WSL), Terry Weissman (DEC-UEG-WSL), Edward Moy (Berkeley), Ralph R. Swick (MIT-Athena), Mark Vandevoorde (MIT-Athena), Bob McNamara (DEC-MAD), Jim Gettys (MIT-Athena), Bob Scheifler (MIT X Consortium), Doug Mink (SAO), Steve Pitschke (Stellar), Ron Newman (MIT-Athena), Jim Fulton (MIT X Consortium), Dave Serisky (HP), and Jonathan Kamens (MIT-Athena).

Name

`xtici` – TekColor™ editor.

Syntax

`xtici` [*options*]

Description

`xtici` (also known as the TekColor Editor) allows you to create precise colors using formats that are portable from system to system. The valid color formats are called *color spaces*. `xtici` was written by Chuck Adams and Al Tabayoyon of Tektronix, Inc., to take advantage of the powers of the X Color Management System (Xcms), also developed by Tektronix and donated to the X Consortium as part of X11R5. The TekColor Editor (`xtici`) is available as a public domain client.

Chapter 8, *Other Clients*, describes how to use `xtici`. See Chapter 12, *Specifying Color*, for an overview of the Xcms color model.

Options

`-display` [*host*]:*server*[.*screen*]

Specifies the name of the display to use. *host* is the hostname of the physical display, *server* specifies the display server number, and *screen* specifies the screen number. Either or both of the *host* and *screen* elements to the display specification can be omitted. If *host* is omitted, the local display is assumed. If *screen* is omitted, screen 0 is assumed (and the period is unnecessary). The colon and (`display`) *server* are necessary in all cases.

For example:

```
% xtici -display your_node:0.1
```

specifies the screen 0 on server 0 on the display identified by *your_node*. If the host is omitted, the local display is assumed. If the screen is omitted, the screen 0 is assumed; the server and colon (`:`) are necessary in all cases.

The `-display` option can be abbreviated as `-d`, unless the client accepts another option that begins with “d.”

`-gamutProc` *state*

The acceptable values are `closest`, `chroma`, and `value`.

`-huebar` *state*

Specifies the appearance and behavior of the hue bar. The acceptable values are `empty`, `constant`, and `dynamic`.

`-leaf` *appearance*

Specifies the appearance of the hue leaf. The acceptable values are `filled` and `empty`.

Resources

xtici is written using the Athena widget set. See “Widget Hierarchy” for a diagram of the widgets and Appendix G, *Widget Resources*, for a list of resources.

Widget Hierarchy

```

TekHVC xtici
    TekBox main
        Form menubar
            Command quit
            Command option
                TransientShell optionmenu
                    Form menuform
                        Command huebutton
                        Command leafbutton
                        Command clamp
                        Command coordinates
                            TransientShell coordmenu
                                Form menuform
                                    Command rgb
                                    Command uvY
            Command import
            Command export
            Command edit
                TransientShell editmenu
                    Form menuform
                        Command copy
                            TransientShell copymenu
                                Form menuform
                                    Command copyhvc
                                    Command copyrgb
                                    Command copyuvy
                        Command paste
            Command help
                TransientShell helpmenu
                    Form menuform
                        Command interface
                        Command version
                        Command quit
                        Command option
                        Command import
                        Command export
                        Command edit
    ColorScale scale
        Label label
        RepeaterButton up
        RepeaterButton down
        Scrollbar scrollbar
        Colorbar bar

```

```

Zoom zoom
Colorbar expand
Form show
  TriText hvc
    Form form
      Label item1
      Label item2
      Label item3
      Text text1
      Text text2
      Text text3
  TriText uvy
    Form form
      Label item1
      Label item2
      Label item3
      Text text1
      Text text2
      Text text3
  TriText rgb
    Form form
      Label item1
      Label item2
      Label item3
      Text text1
      Text text2
      Text text3
Form patch
  Text patchtext
  Box patcharea
ColorScale huebar
  Label label
  RepeaterButton up
  RepeaterButton down
  Scrollbar scrollbar
  Colorbar bar
  Zoom zoom
  Colorbar expand
Leaf leaf
  Form hvcform
    Label value
    RepeaterButton uparrow
    RepeaterButton downarrow
    Hueleaf hueleaf
    RepeaterButton leftarrow
    RepeaterButton rightarrow
    Label chroma

```

See Also

xcol, xcoloredit; Chapter 8, *Other Clients*; Chapter 12, *Specifying Color*.

Trademarks

Tektronix® and Tek® are registered trademarks of Tektronix, Inc.

TekColor™ Color Management System, TekColor™ CMS, TekColor™ Human Interface, and TekHVC Color Space are trademarks of Tektronix, Inc.

Copyright

Copyright 1991, Tektronix, Inc.

Author

Chuck Adams, Al Tabayoyon, Tektronix, Inc.

Name

`xwd` – place window images in a dump file.

Syntax

`xwd` [*options*]

Description

`xwd` stores window images in a specially formatted window dump file. This file can then be read by various other X utilities for redisplay, printing, editing, formatting, archiving, image processing, etc. The target window is selected by clicking the pointer in the desired window. The keyboard bell is rung once at the beginning of the dump and twice when the dump is completed.

Options

`xwd` accepts the following options:

`-add value`

Specifies a signed value to be added to every pixel.

`-display [host]:server[.screen]`

Allows you to specify the host, server, and screen to connect to. *host* is the host-name of the physical display, *server* is the server number, and *screen* is the screen number. For example,

```
xwd -display your_node:0.1 &
```

specifies screen 1 on server 0 on the display named by *your_node*. Either or both of the *host* and *screen* elements to the display specification can be omitted. If *host* is omitted, the local display is assumed. If *screen* is omitted, screen 0 is assumed (and the period is unnecessary). The colon and (display) *server* are necessary in all cases.

`-frame`

Indicates that the window manager frame should be included when manually selecting a window.

`-help` Prints out the “Usage:” command syntax summary.

`-icmap`

Normally the colormap of the chosen window is used to obtain RGB values. This option forces the first installed colormap of the screen to be used instead. (Available as of Release 5.)

`-id window_ID`

Allows you to specify the window using its *window_ID* (resource ID), rather than by selecting it with the pointer.

- name** *name*
Allows you to specify the window using its *name* (stored in the WM_NAME property), rather than by selecting it with the pointer.
- nobdrs**
Specifies that the window dump should not include the pixels that compose the X window border. This is useful when the window contents are to be included in a document as an illustration.
- out** *file*
Allows you to specify the output file on the command line. The default outputs to the standard output (*stdout*).
- root** Makes a dump of the entire root window; the user is not required to select a window with the pointer.
- screen**
Indicates that the `GetImage` request used to obtain the image should be done on the root window, rather than directly on the specified window. In this way, you can obtain pieces of other windows that overlap the specified window, and more importantly, you can capture menus or other popups that are independent windows but appear over the specified window. (Available as of Release 5.)
- xy** Applies to color displays only. The `-xy` option selects “XY” pixmap format dumping instead of the default “Z” pixmap format.

Files

XWDFile.h

X Window Dump File format definition file.

See Also

X, xdpr, xpr, xwud.

Author

Tony Della Fera, Digital Equipment Corp., MIT Project Athena;
William F. Wyatt, Smithsonian Astrophysical Observatory.

Name

xwininfo – window information utility for X.

Syntax

xwininfo [*options*]

Description

xwininfo (described in Chapter 8, *Other Clients*) is a utility for displaying information about windows. Depending on which options are chosen, various information is displayed. If no options are chosen, `-stats` is assumed.

The user has the option of selecting the target window either by using the pointer or by specifying the window on the command line. To select the window using the pointer, simply click any pointer button in the desired window. There are two ways to specify the target window on the command line: either by window ID (using the `-id` option); or by name (using the `-name` option). See “Options” below. There is also a special `-root` option to obtain information about the root window.

With the `-children` and `-tree` options (Release 5), *xwininfo* can now provide information that, in prior releases, was only available using the *xlswins* client. *xlswins* has been removed from the standard MIT X distribution in Release 5.

Options

xwininfo accepts the following options:

- `-all` A quick way to ask for all information possible.
- `-bits` Causes the display of various attributes pertaining to the selected window’s raw bits and how the selected window is to be stored. Information displayed includes the selected window’s bit gravity, window gravity, backing store hint, backing planes value, backing pixel, and whether or not the window has save under set.
- `-children`
Displays the selected window’s root, parent, and children windows’ IDs and names. (Available as of Release 5.) See `-tree`.
- `-display [host]:server[.screen]`
Allows you to specify the host, server, and screen to connect to. *host* is the host-name of the physical display, *server* specifies the server number, and *screen* specifies the screen number. For example,

```
% xwininfo -display your_node:0.1
```

specifies screen 1 of server 0 on the display hardware named by *your_node*. If the host is omitted, the local display is assumed. If the screen is omitted, screen 0 is assumed; the server and colon (:) are necessary in all cases.

- events**
Causes the selected window's event masks to be displayed. Both the event mask of events wanted by some client and the event mask of events not to propagate are displayed.
- english**
Causes all individual height, width, and x and y positions to be displayed in inches (and feet, yards, and miles if necessary), as well as number of pixels, based on what the server thinks the resolution is. Geometry specifications that are in *+x+y* form are not changed. **-metric** and **-english** may be used at the same time.
- frame**
Causes window manager frames to be considered when manually selecting windows.
- help** Prints out the "Usage:" command syntax summary.
- id *window_ID***
Allows the user to specify a target window on the command line by providing its *window_ID* (rather than selecting a window using the pointer). This is very useful in debugging X applications where the target window is not mapped to the screen or where the use of the pointer might be impossible or interfere with the application.
- int** Specifies that all X window IDs should be displayed as integer values. The default is to display them as hexadecimal values.
- metric**
Causes all individual height, width, and x and y positions to be displayed in millimeters, as well as number of pixels, based on what the server thinks the resolution is. Geometry specifications that are in *+x+y* form are not changed. **-english** and **-metric** may be used at the same time.
- name *name***
Allows the user to specify a target window on the command line by providing its *name* (rather than selecting a window using the pointer).
- root** Specifies that the root window is the target window. This is useful in situations where the root window is completely obscured.
- shape**
Causes the selected window's window and border shape extents to be displayed. (Available as of Release 5.)
- size** Causes the selected window's sizing hints to be displayed. Information displayed includes: for both the normal size hints and the zoom size hints, the user supplied location, if any; the program supplied location, if any; the user supplied size, if any; the program supplied size, if any; the minimum size, if any; the maximum size, if any; the resize increments, if any; and the minimum and maximum aspect ratios, if any.

- stats Causes various attributes of the selected window having to do with its location and appearance to be displayed. Information displayed includes the location of the window, its width, height, depth, border width, class, and map state, colormap ID (if any), backing store hint, and the location of its corners. If *xwininfo* is run with no options, *-stats* is assumed.
- tree Like *-children* but displays all children recursively. (Available as of Release 5.)
- wm Causes the selected window's window manager hints to be displayed. Information displayed may include whether or not the application accepts input, what the window's icon window number and name is, where the window's icon should go, and what the window's initial state should be.

Examples

The following is sample output taken with no options specified. (The Motif window manager was running on the display. See Chapter 8, *Other Clients*, for a discussion of how the *mwm* frame can affect *xwininfo*'s results.)

```
xwininfo: Please select the window about which you
          would like information by clicking the
          mouse in that window.
```

```
xwininfo: Window id: 0x3c0000f "xterm"
```

```
Absolute upper-left X: 8
Absolute upper-left Y: 25
Relative upper-left X: 0
Relative upper-left Y: 0
Width: 819
Height: 484
Depth: 8
Visual Class: PseudoColor
Border width: 0
Class: InputOutput
Colormap: 0x27 (installed)
Bit Gravity State: NorthWestGravity
Window Gravity State: NorthWestGravity
Backing Store State: NotUseful
Save Under State: no
Map State: IsViewable
Override Redirect State: no
Corners: +8+25 -325+25 -325-391 +8-391
-geometry 80x24+0+0
```

Bugs

Using `-stats` and `-bits` together shows some redundant information.

The geometry string displayed must make assumptions about the window's border width, the behavior of the application, and the window manager. As a result, the location given is not always correct.

See Also

X, `xprop`; Chapter 8, *Other Clients*.

Author

Mark Lillibridge, MIT Project Athena.

Name

xwud – X window image displayer.

Syntax

xwud [*options*]

Description

xwud is an X Window System window image undumping utility. *xwud* allows X users to display a window image saved in a specially formatted dump file, such as one produced by *xwd*. Chapter 8, *Other Clients*, describes how to use these clients.

xwud allows you to specify the coordinates at which this image is displayed using the `-geometry` option. By default, *xwud* displays the window image at the coordinates of the original window from which the dump was taken.

Options

xwud accepts the following options:

`-bg color`

If a bitmap image (or a single plane of an image) is displayed, this option can be used to specify the color to display for the “0” bits in the image.

`-display [host]:server[.screen]`

Allows you to specify the host, server, and screen to connect to. *host* is the host-name of the physical display, *server* specifies the server number, and *screen* specifies the screen number. For example:

```
% xwud -display your_node:0.1
```

specifies screen 1 on server 0 on the display named by *your_node*. If the host is omitted, the local machine is assumed. If the screen is omitted, the screen 0 is assumed; the server and colon (:) are necessary in all cases.

`-fg color`

If a bitmap image (or a single plane of an image) is displayed, this option can be used to specify the color to display for the “1” bits in the image.

`-geometry geometry`

The *xwud* window is created with the specified size and location determined by the supplied geometry specification. The `-geometry` option can be (and often is) abbreviated to `-g`, unless there is a conflicting option that begins with “g.” The argument to the geometry option (*geometry*) is referred to as a “standard geometry string,” and has the form *widthxheight±xoff±yoff*. (This option is available for use with *xwud* as of Release 4.)

Typically, you will want to specify only the position and let the size default to the actual size of the image.

`-help` Prints out a short description of the allowable options.

- in *file***
Allows the user to specify the input file on the command line. If no file is specified, standard input is assumed.
- new** Forces creation of a new colormap for displaying the image. If the image characteristics happen to match those of the display, this can get the image on the screen faster, but at the cost of using a new colormap (which on most displays will cause other windows to go technicolor).
- noclick**
Clicking any button in the window will terminate the application, unless this option is specified. Termination can always be achieved by typing “q”, “Q”, or Control-c.
- plane *number***
Selects a single bit plane of the image to display. Planes are numbered with zero being the least significant bit. This option can be used to figure out which plane to pass to *xpr* for printing.
- raw** Forces the image to be displayed with whatever color values happen to currently exist on the screen. This option is mostly useful when undumping an image back onto the same screen that the image originally came from, while the original windows are still on the screen, and results in getting the image on the screen faster.
- rv** If a bitmap image (or a single plane of an image) is displayed, this option forces the foreground and background colors to be swapped. This may be needed when displaying a bitmap image which has the color sense of pixel values “0” and “1” reversed from what they are on your display.
- std *map_type***
Causes the image to be displayed using the specified standard colormap. The property name is obtained by converting the type to uppercase, prepending “RGB_”, and appending “_MAP”. Typical types are best, default, and grey. See *xstdcmap* for one way of creating standard colormaps.
- vis *vis_type_or_ID***
Allows you to specify a particular visual or visual class. The default is to pick the “best” one. A particular class can be specified: StaticGray, GrayScale, StaticColor, PseudoColor, DirectColor, or TrueColor. Or Match can be specified, meaning use the same class as the source image. Alternatively, an exact visual ID (specific to the server) can be specified, either as a hexadecimal number (prefixed with “0x”) or as a decimal number. Finally, default can be specified, meaning use the same class as the colormap of the root window. Case is not significant in any of these strings.

Window Image Displayer

xwud *(continued)*

Files

XWDFile.h

X Window Dump File format definition file.

See Also

X, xdpr, xpr, xstdcmap, xwd; Chapter 8, *Other Clients*.

Author

Bob Scheifler, MIT X Consortium;

Part Four:

Appendices

This part of the book contains useful reference information.

Managing Your Environment
Release 5 Standard Fonts
Standard Bitmaps
Standard Cursors
xterm Control Sequences
Translation Table Syntax
Widget Resources
Obtaining Example Programs
Glossary
Index

A

Managing Your Environment

This appendix discusses various tasks involved in managing your X user environment, mostly from the UNIX point of view.

In This Appendix:

Including X in Your Search Path	743
A Startup Shell Script	744
What Should Go in the Script	744
Server Access Control	748
Host-based Access and the xhost Client	748
User-based Access: xdm and the .Xauthority File	749
Console Messages	750

A

Managing Your Environment

Throughout this guide, we've demonstrated various things you can do to tailor your X user environment to your needs. This appendix explains some additional tasks you might need (or want) to perform in order to keep your user environment running smoothly. We'll take a look at:

- Including X in your search path
- Writing a startup shell script
- Addressing security issues and access control
- Redirecting console messages

X exists in so many incarnations and runs on so many different versions of UNIX (not to mention other operating systems) that it is difficult to be definitive about these tasks. The current appendix assumes you are running a fairly standard version of X with UNIX. However, given the various incarnations of both X and UNIX, you should be sure to check your system's documentation for additional (or contrary) details.

For a broader and more substantial discussion of these and other system issues, see Volume Eight, *X Window System Administrator's Guide*.

Including X in Your Search Path

The various X clients are normally stored in the directory */usr/bin/X11*. In order to invoke them by name like any other UNIX program, you need to make this directory part of your search path.

This is normally done from your *.cshrc* (C shell) or *.profile* (Bourne shell) file, using a command similar to this:

Bourne Shell:

```
PATH=/usr/ucb:/bin:/usr/bin:/usr/bin/X11:/usr/local/bin
export PATH
```

C Shell:

```
set path=( /usr/ucb /bin /usr/bin /usr/bin/X11 /usr/local/bin )
```

The exact list of directories will differ from system to system. Be aware that directories are searched sequentially from left to right, so a command with the same name in an earlier directory will be found and used before one in a later directory. Many users take advantage of this fact to run customized versions of programs by putting "." (the current directory) or a local tools directory first in their search path. This works fine but you should be aware that this provides a security loophole that can be taken advantage of by an experienced system cracker. It's much safer to put a period at the end of your path, or eliminate it entirely.

If you have already logged in before adding the above line to your *.profile* or *.cshrc* file, you should log out and log in again, or type in the path-setting command at your prompt, so that it takes effect for your current session.

A Startup Shell Script

It's a basic principle of UNIX to let the computer do the work. Accordingly, you'd no doubt like to run various X clients automatically whenever you log in.

The best way to do this is to create a script that runs the clients you want. Depending on how X is set up on your system, you can execute this script in one of two ways:

- If *xdm* is running X, name the script *.xsession*, make it executable, and put it in your home directory. When you log in, *xdm* will automatically execute your *.xsession* file. *.xsession* is generally a Bourne shell script, but it can also be a C shell script or any other executable. If *\$HOME/.xsession* doesn't exist, *xdm* will give you an *xterm* client and start the *twm* window manager.
- If you are starting X either with *xinit* or with the *startx* shell script (which is a front-end to *xinit*), name the script *.xinitrc* and put it in your home directory. Both *xinit* and *startx* starts the server and executes *.xinitrc*. Unlike *.xsession*, the *.xinitrc* script must be a Bourne shell script and does not have to be executable. Among the extra functionality provided by *startx* is that if *\$HOME/.xinitrc* doesn't exist, *startx* defaults to the system-wide startup script */usr/lib/X11/xinit/xinitrc*, whereas *xinit* will just give you a single *xterm* client.

What Should Go in the Script

With some variation depending on the specific environment, in most cases your startup script should do the following:

- Load your resources file with *xrdb*.
- Set up other server preferences, such as bell volume, font path, etc.
- Start the window manager.
- Start other clients you want on your default display, such as *xterm*, *oclock*, *xload*, etc.
- Run an *xterm* process in the foreground; terminating this process will terminate the login session.

The sample script appears as Example A-1. You can use this script as either an *.xsession* or *.xinitrc* script.

One thing you may notice in the startup script is that some commands are run in the background and some aren't. For clients that configure the X server like *xrdb*, you want to be sure that the command completes before other clients are started (to make sure that the other clients can retrieve all the right resources). Since *xrdb* exits as soon as the resources are successfully loaded, this works great. For clients like *xterm* and *xclock*, however, the client remains active until they are explicitly exited, so you need to start them in the background or else they will prevent the script from continuing.

The exception is the last command, which should be left in the foreground. The reason is that the X session only remains active as long as the startup script is active. You don't want the startup script to actually exit, since then the X session will terminate—what you will see is that all your windows appear and then immediately disappear. Leaving the last command in the foreground is therefore crucial to the longevity of your X session. The last command is often called the *controlling process*, since it is the only thing preventing the startup script and X session from exiting. We suggest using an *xterm* window for the controlling process, but some people use the window manager.

If you use an *xterm* window for the controlling process, you have to be careful not to exit that shell accidentally. For that reason, we use several precautions to make sure that we don't kill that window. We start the window iconified, we start it in reverse video, and we make sure it's properly labeled so that we always know which window is the controlling process. In addition, if you use a shell with an "autologout" feature as your controlling process, you should make sure it isn't in effect for the controlling *xterm* window: otherwise, you might be typing furiously in another window, but if you haven't typed in the controlling *xterm* window for a while then you might be logged out of your entire X session.

Another thing to note in the sample script is that we set up a separate display name for remote clients to use. As explained in Chapter 3, *Working in the X Environment*, clients running on the local machine access the `DISPLAY` variable to determine on which physical display to create windows. Without explicit settings, both *xdm* and *xinit* will automatically set `DISPLAY` properly on a workstation, to either `unix:0.0` or `:0.0`, and *xdm* will set `DISPLAY` properly for an X terminal as well (e.g., `ncd5.ora.com:0.0`).

The `DISPLAY` environment variable, however, is not propagated to remote shells. And even if it were, you wouldn't always want it to be. When running a client on a remote machine, you have to explicitly use the `-display` command-line option to the remote client to tell it what display to use. For a workstation, you have to be careful that you use the right display name. You can't just use the value of `DISPLAY`, since by default it's set to `unix:0.0` or `:0.0`, and if it's used on a remote machine then the client will attempt to display to the local display server of the remote machine, not yours. Instead, you have to make sure that the display name supplied to the remote client follows the `hostname:0.0` form.

The *rsh* command also requires that you have set up the remote system to accept commands from the local host. This means that you need a file in your home directory on the remote host called *.rhosts* containing the name of the local system.

Example A-1 shows a startup Bourne shell script which would open windows on the display as shown in Figure A-1.

Example A-1. Startup Bourne shell script for a workstation

```
#!/bin/sh
# Get hostname.
cpu=`hostname`
# If on a workstation, DISPLAY is set to :0.0 or unix:0.0. If you
# want to run remote clients, you need to use hostname:0.0. Set that
# up:
case $DISPLAY in
  unix:0.0|unix:0|:0.0|:0) REMOTEDISPLAY="$cpu:0.0";;
  *) REMOTEDISPLAY=$DISPLAY ;;
esac
# Load resource definitions from .Xresources
xrdb $HOME/.Xresources
# Set keyclick off and invoke the screen saver after
# seven minutes of idleness
xset c off s 420
# Start the mwm window manager
mwm &
# xconsole window will disappear if you do not log in at the console
xconsole -exitOnFail -geometry +0+0 &
# Now start up some xterms
# Start an xterm near lower-left corner but above icons
xterm -geometry 80x22+0-100 &
# Place an xterm next to it
xterm -geometry 98x22-10-10 &
# remote xterm above (but below xconsole window)
rsh ruby xterm -name RUBY -geometry 80x25+0+110 -display $REMOVEDISPLAY &
# Now start up other clients
# digital xclock in upper-right corner
xclock -digital -update 1 -geometry -0+0 &
# xcalc just below it;
xcalc -geometry -0+75 &
# xload at bottom of xcalc
xload -geometry -0+350 &
# Start another xterm window.
# This is the only xterm that should be run in the foreground.
# Killing this window will shut down your X session.
exec xterm -iconic -rv -name "LOGOUT"
```

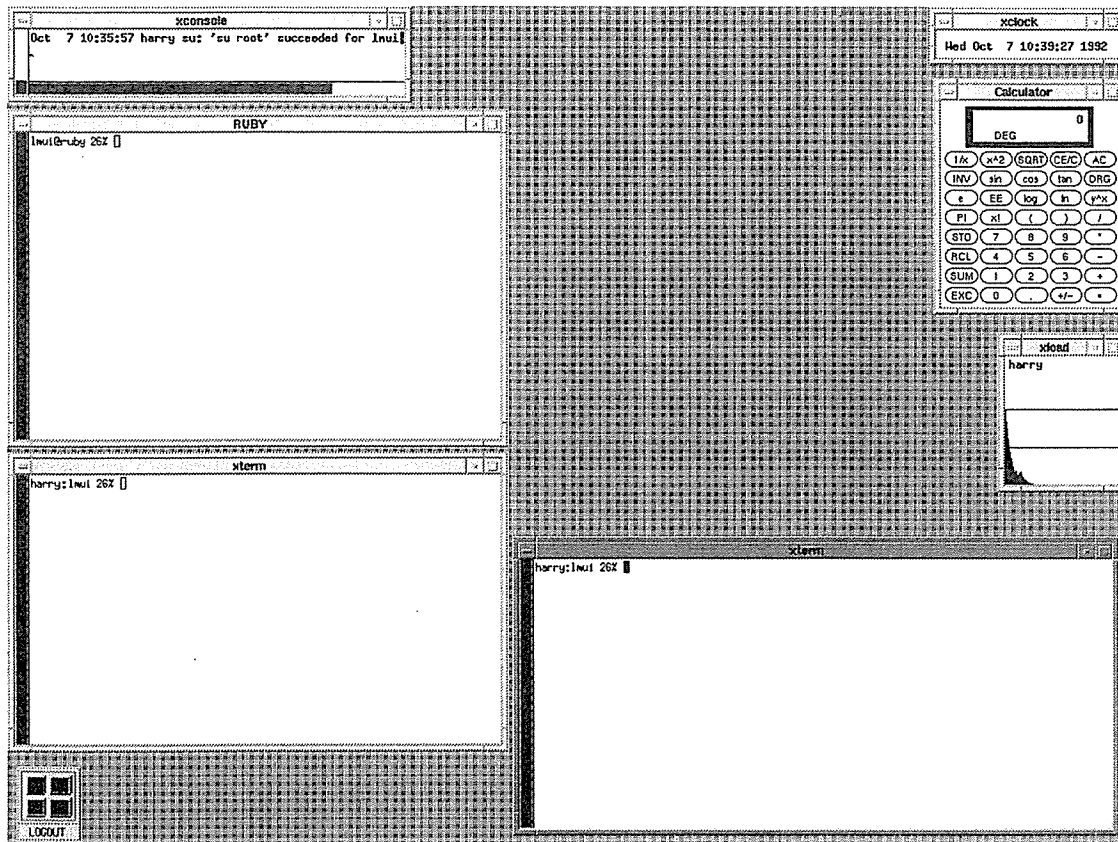


Figure A-1. Display after running either sample script

Note that windows are actually arranged in a “tiled” fashion, with two large xterm windows side by side on the bottom of the screen, a smaller one (connected to a remote system) above, and the “desk accessories” lined up in the upper-right corner. This leaves some room free for new windows or for invoking the Root Menu menu on the root window. This is ideal for our purposes, which are mainly editing, formatting, and testing examples for books. Depending on what you do, another arrangement might be better.*

The startup script calls a *xconsole* client, which has been added to the standard distribution in Release 5. *xconsole* can only be run by someone who actually logs in at the console display. Thus, although this startup script can be used for both workstations and X terminals, X terminals will not be able to receive the *xconsole* window unless that user is also logged in at the console.

* This startup script was developed for and run on a Sun workstation. Differences in pixel sizes and resource definitions may make the coordinates and sizes of various windows come out differently on other hardware.

Server Access Control

X is designed so that clients can connect to the server from any other host on the network. This poses some security problems, since it means that other users on other hosts might be able to access your server. As a solution, X provides mechanisms for restricting access to your server. These are:

- Host-based access control. Under host-based access control, the X server only accepts connections from a specified list of hosts. The list of hosts is specified in the file */etc/X0.hosts* and can also be supplemented using the *xhost* client.
- MIT-MAGIC-COOKIE-1. This scheme involves a special code called a “magic cookie” which is known by the X server and is also made available to the user’s account. Clients need to be able to present the magic cookie before they can access the server.
- XDM-AUTHORIZATION-1. This is similar to MIT-MAGIC-COOKIE-1, but the code is encrypted so that it can’t be snooped as it is passed over the network.
- SUN-DES-1. This method uses Secure RPC.

The common methods for restricting access to your server are host-based access control and the MIT-MAGIC-COOKIE-1 scheme, which is also called user-based access control. The XDM-AUTHORIZATION-1 and SUN-DES-1 are also user-based methods, but they are only available with X11R5. Furthermore, since they are built upon DES (Data Encryption Standard), they are not exportable outside of the U.S. For complete information on any of these schemes, see Volume Eight, *X Window System Administrator’s Guide*.

Host-based Access and the xhost Client

The */etc/X0.hosts* file contains a list of systems that are allowed to access the server. The “0” stands for the number of your server—if you ran a second server on your machine (that is, a server with the display name *hostname:1.0*), you would use */etc/X1.hosts*. In most cases, you only have one server running on your machine, so */etc/X0.hosts* is the only file you need to worry about.

By default, */etc/X0.hosts* does not exist, and only clients run on the local host can connect to the server. You should create and edit this file so that it contains the list of systems you want to have access to your server on a regular basis.

The *xhost* client can be used to give (or deny) systems access to the server interactively, possibly overriding the contents of */etc/X0.hosts*. Specifying a host name with an optional leading plus sign (+) allows the host to access the server, and specifying a host name with a leading minus sign (–) prevents a previously allowed host from accessing the server. Multiple hosts can be specified on the same line. Running *xhost* without any arguments prints the current hosts allowed to access your display.

For example, to add the hosts *jupiter* and *saturn*, and remove *neptune*:

```
% xhost +jupiter saturn -neptune
```

To be truly secure, you might use the fully qualified domain names. That way another machine called jupiter in another domain won't be able to access your server as well. Use fully qualified domain names in the */etc/X0.hosts* and *.rhosts* files as well.

```
% xhost +jupiter.ora.com saturn.ora.com -neptune.camb.com
```

Note that when a remote system is denied access to your server, no one on that system can display clients to your server, including yourself. Of course, the reverse is also true: when you allow yourself to run a client from a given host, you also allow everyone else on that host.

This is the main problem with host-based access control. If you have the only account on your workstation then there isn't a problem, but if you have NIS (aka Yellow Pages) running, then it's likely that all other users at your site have accounts on your machine. Which means that despite how carefully you use host-based access control, it doesn't provide any protection against that devious prankster across the hall.

User-based Access: xdm and the .Xauthority File

As of Release 4, the display manager and its control protocol (XDMCP) provide a user-based access control mechanism, which can be used to supplement or replace the host-based access mechanism discussed in the previous section. Release 4 and Release 5 *xdm* can be set up to provide user authorization on a particular display. By default, authorization is enabled for the local console display (:0) and is disabled for other displays managed by *xdm*, such as X terminals. To enable user-based access control for X terminals, set the following resource in the */usr/lib/X11/xdm/xdm-config* file:

```
DisplayManager*authorize: true
```

In the default *xdm-config*, this resource is set to *false*.

If authorization is enabled, then when you log in, *xdm* places a machine-readable access code, known as a magic cookie, in a file called *.Xauthority* in your home directory. *xdm* also makes this magic cookie available to the server.

The magic cookie defined in a user's *.Xauthority* file is basically a secret code shared by the server and a particular user logged in on a particular display. When a client requests access to the server, the server checks to see whether the client program has access to the magic cookie. All processes started by the user in question have that access, and thus the server allows access to that user's clients. Basically, under the magic cookie authorization scheme, a display becomes user-controlled.

The security mechanism provided by the magic cookie is evident in a situation in which another user tries to run a client on your machine. The server requires the client run by the other user to have access to the magic cookie shared exclusively between you and the server. The other user cannot provide the proper authorization code and thus cannot run a client on your host.

Of course, in many cases, users in a network will want to run clients on several machines (while displaying the client window on their local displays). This can be done if a user supplies authorization information associated with his local server to the remote host. The *xauth*

client allows users to transfer this information to the remote machine. Basically, *xauth* is a utility to manipulate *.Xauthority* files.

The most common use for *xauth* is to extract a user's authorization information for the current display, copy it to another machine, and merge it into the server's authorization records on the remote machine like this:

```
% xauth extract - $DISPLAY | rsh host2 xauth merge -
```

The dash (-) arguments indicate that extracted authorization records should be written to the standard output and that the *xauth* merge function should accept records from standard input. This command supplies the remote server with authorization information, allowing the user to run a remote shell on that host. See the *xauth* reference page in Part Three of this guide for more information. Note that this command line depends on the *.rhosts* file on the remote machine being set up to allow remote commands from the local machine.

If an installation is using remote file sharing, such as NFS, then sharing authorization records may not be an issue. If every user has a single home directory that is accessible to all machines, the machines have access to the necessary *.Xauthority* files at all times. In such an environment, users should be able to run programs on any of the networked machines without using *xauth*.

Host-based access control overrides user-based access control. That is, if you have added a host to your access control list, then all users on that host can access your server regardless of whether you use the magic cookie as well.

Console Messages

On a single-user workstation, it is likely that the screen used for running X is also used as the system console. Console messages from the kernel may appear on the screen, overlaying the X windows. They make a nasty mess of the screen. You can refresh the display and erase the console message by running the *xrefresh* client (described in Chapter 8, *Other Clients*).

However, Release 5 offers a better solution. The *xconsole* client is intended to avoid the problem of messages obscuring the screen altogether by providing a small window in which the */dev/console* messages are displayed. The *xconsole* process should be run early in the startup script so that any messages generated during startup are "captured." A console window is shown in Figure A-1. For more information, see the *xconsole* reference page in Part Three of this guide.

Note that some implementations of X support a *-C* option to *xterm* that redirects messages sent to */dev/console* to that *xterm* window. If this option is supported, you *can* add the *-C* option to the console *xterm* in your startup file. After this window is mapped (displayed on the screen), all such messages are displayed there. However, it is generally preferable to use *xconsole*.

B

Release 5 Standard Fonts

This appendix shows the standard display fonts available in Release 5 of the MIT X distribution. The images contained in this appendix are window dumps created with our own program, called xshowfonts, the code for which is included.

B

Release 5 Standard Fonts

This appendix includes pictures of some representative fonts from the standard X distribution in Release 5. Not every font may be supported by particular server vendors, and some vendors may supplement the set.

The standard fonts are stored in four directories. The first three directories contain bitmap fonts; the *Speedo* directory contains outline fonts. See Chapter 6, *Font Specification*, for more information.

Directory	Contents
<i>/usr/lib/X11/fonts/misc</i>	Six fixed-width fonts, the cursor font, other miscellaneous fonts.
<i>/usr/lib/X11/fonts/75dpi</i>	Fixed- and variable-width fonts, 75 dots per inch.
<i>/usr/lib/X11/fonts/100dpi</i>	Fixed- and variable-width fonts, 100 dpi.
<i>/usr/lib/X11/fonts/Speedo</i>	Charter and Courier outline fonts from Bitstream.

Tables B-1 through B-4 list the fonts in each of the four Release 5 font directories. The first column lists the name of the file in which the font is stored (without the *.pcf* extension); the second column lists the actual font name. See Chapter 6, *Font Specification*, for information about font naming conventions.

PICTURES of the different font families supplied in the MIT X11 distribution appear on subsequent pages. We show just the fonts in the *75dpi* directory. The *100dpi* directory contains the same fonts stored in the *75dpi* directory but for 100 dots per inch monitors. Keep in mind that all of the fonts in the *75dpi* and *100dpi* directories are available in 8-, 10-, 12-, 14-, 18-, and 24-point sizes. Each page shows fonts of various sizes, weights, and styles. We include the source for *xshowfonts.c*, the program we wrote to make these displays, at the end of the appendix.* We also show you, using *xfd*, one example of each of the unique character sets available.

All of the characters in each font in the *75dpi* directory are shown actual size, as they would appear on a 900 × 1180 pixel, 10" × 13.5" screen (Sun). On a screen with different pixel density, these fonts would appear in a different size.

*If you don't want to type this program in, you can obtain the source from uunet.uu.net via anonymous *ftp* or *uucp*. See Appendix H, *Obtaining Example Programs*, for more information.

Fonts that begin with many blank characters are shown with most leading blanks removed. Therefore, you can't always get the character number of each cell in the font by counting from the first cell we have shown. Use *xfd* to quickly determine the code for a particular cell.

Table B-1. Fonts in the misc Directory

Filename	Font name
6x12.pcf	-misc-fixed-medium-r-semicondensed--12-110-75-75-c-60-iso8859-1
6x13.pcf	-misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso8859-1
6x10.pcf	-misc-fixed-medium-r-normal--10-100-75-75-c-60-iso8859-1
7x13.pcf	-misc-fixed-medium-r-normal--13-120-75-75-c-70-iso8859-1
7x14.pcf	-misc-fixed-medium-r-normal--14-130-75-75-c-70-iso8859-1
clR8x12.pcf	-schumacher-clean-medium-r-normal--12-120-75-75-c-80-iso8859-1
clR8x13.pcf	-schumacher-clean-medium-r-normal--13-130-75-75-c-80-iso8859-1
6x9.pcf	-misc-fixed-medium-r-normal--9-90-75-75-c-60-iso8859-1
clR8x10.pcf	-schumacher-clean-medium-r-normal--10-100-75-75-c-80-iso8859-1
5x7.pcf	-misc-fixed-medium-r-normal--7-70-75-75-c-50-iso8859-1
clR8x16.pcf	-schumacher-clean-medium-r-normal--16-160-75-75-c-80-iso8859-1
clR8x14.pcf	-schumacher-clean-medium-r-normal--14-140-75-75-c-80-iso8859-1
clR8x8.pcf	-schumacher-clean-medium-r-normal--8-80-75-75-c-80-iso8859-1
5x8.pcf	-misc-fixed-medium-r-normal--8-80-75-75-c-50-iso8859-1
clR9x15.pcf	-schumacher-clean-medium-r-normal--15-150-75-75-c-90-iso8859-1
clR6x8.pcf	-schumacher-clean-medium-r-normal--8-80-75-75-c-60-iso8859-1
clR5x6.pcf	-schumacher-clean-medium-r-normal--6-60-75-75-c-50-iso8859-1
clR7x8.pcf	-schumacher-clean-medium-r-normal--8-80-75-75-c-70-iso8859-1
clR4x6.pcf	-schumacher-clean-medium-r-normal--6-60-75-75-c-40-iso8859-1
clR5x8.pcf	-schumacher-clean-medium-r-normal--8-80-75-75-c-50-iso8859-1
clR6x6.pcf	-schumacher-clean-medium-r-normal--6-60-75-75-c-60-iso8859-1
6x13B.pcf	-misc-fixed-bold-r-semicondensed--13-120-75-75-c-60-iso8859-1
12x24rk.pcf	-sony-fixed-medium-r-normal--24-170-100-100-c-120-jisx0201.1976-0
7x13B.pcf	-misc-fixed-bold-r-normal--13-120-75-75-c-70-iso8859-1
7x14B.pcf	-misc-fixed-bold-r-normal--14-130-75-75-c-70-iso8859-1
clR6x12.pcf	-schumacher-clean-medium-r-normal--12-120-75-75-c-60-iso8859-1
clR6x13.pcf	-schumacher-clean-medium-r-normal--13-130-75-75-c-60-iso8859-1
clR6x10.pcf	-schumacher-clean-medium-r-normal--10-100-75-75-c-60-iso8859-1
clR7x12.pcf	-schumacher-clean-medium-r-normal--12-120-75-75-c-70-iso8859-1
clR7x10.pcf	-schumacher-clean-medium-r-normal--10-100-75-75-c-70-iso8859-1
clR7x14.pcf	-schumacher-clean-medium-r-normal--14-140-75-75-c-70-iso8859-1
8x13.pcf	-misc-fixed-medium-r-normal--13-120-75-75-c-80-iso8859-1
8x16.pcf	-sony-fixed-medium-r-normal--16-120-100-100-c-80-iso8859-1
clR5x10.pcf	-schumacher-clean-medium-r-normal--10-100-75-75-c-50-iso8859-1
9x15.pcf	-misc-fixed-medium-r-normal--15-140-75-75-c-90-iso8859-1
heb6x13.pcf	-misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso8859-8
clB8x8.pcf	-schumacher-clean-bold-r-normal--8-80-75-75-c-80-iso8859-1
8x13B.pcf	-misc-fixed-bold-r-normal--13-120-75-75-c-80-iso8859-1

Table B-2. Fonts in the misc Directory (continued)

Filename	Font name
7x14rk.pcf	-misc-fixed-medium-r-normal--14-130-75-75-c-70-jisx0201.1976-0
9x15B.pcf	-misc-fixed-bold-r-normal--15-140-75-75-c-90-iso8859-1
clI8x8.pcf	-schumacher-clean-medium-i-normal--8-80-75-75-c-80-iso8859-1
heb8x13.pcf	-misc-fixed-medium-r-normal--13-120-75-75-c-80-iso8859-8
decsess.pcf	decw\$session
clB8x12.pcf	-schumacher-clean-bold-r-normal--12-120-75-75-c-80-iso8859-1
clB8x13.pcf	-schumacher-clean-bold-r-normal--13-130-75-75-c-80-iso8859-1
clB8x10.pcf	-schumacher-clean-bold-r-normal--10-100-75-75-c-80-iso8859-1
clB8x16.pcf	-schumacher-clean-bold-r-normal--16-160-75-75-c-80-iso8859-1
clB8x14.pcf	-schumacher-clean-bold-r-normal--14-140-75-75-c-80-iso8859-1
clB9x15.pcf	-schumacher-clean-bold-r-normal--15-150-75-75-c-90-iso8859-1
olcursor.pcf	-sun-open
hanglg16.pcf	-daewoo-gothic-medium-r-normal--16-120-100-100-c-160-ksc5601.1987-0
8x16rk.pcf	-sony-fixed-medium-r-normal--16-120-100-100-c-80-jisx0201.1976-0
clB6x12.pcf	-schumacher-clean-bold-r-normal--12-120-75-75-c-60-iso8859-1
clB6x10.pcf	-schumacher-clean-bold-r-normal--10-100-75-75-c-60-iso8859-1
jiskan24.pcf	-jis-fixed-medium-r-normal--24-230-75-75-c-240-jisx0208.1983-0
hanglm16.pcf	-daewoo-mincho-medium-r-normal--16-120-100-100-c-160-ksc5601.1987-0
hanglm24.pcf	-daewoo-mincho-medium-r-normal--24-170-100-100-c-240-ksc5601.1987-0
jiskan16.pcf	-jis-fixed-medium-r-normal--16-150-75-75-c-160-jisx0208.1983-0
cursor.pcf	cursor
deccurs.pcf	decw\$cursor
clI6x12.pcf	-schumacher-clean-medium-i-normal--12-120-75-75-c-60-iso8859-1
olgl19.pcf	-sun-open
olgl12.pcf	-sun-open
olgl10.pcf	-sun-open
k14.pcf	-misc-fixed-medium-r-normal--14-130-75-75-c-140-jisx0208.1983-0
olgl14.pcf	-sun-open
nil2.pcf	-misc-nil-medium-r-normal--2-20-75-75-c-10-misc-fontspecific
10x20.pcf	-misc-fixed-medium-r-normal--20-200-75-75-c-100-iso8859-1
12x24.pcf	-sony-fixed-medium-r-normal--24-170-100-100-c-120-iso8859-1



Table B-3. Fonts in the 75dpi Directory

Filename	Font name
courBO10.pcf	-adobe-courier-bold-o-normal--10-100-75-75-m-60-iso8859-1
courBO12.pcf	-adobe-courier-bold-o-normal--12-120-75-75-m-70-iso8859-1
courBO14.pcf	-adobe-courier-bold-o-normal--14-140-75-75-m-90-iso8859-1
courBO18.pcf	-adobe-courier-bold-o-normal--18-180-75-75-m-110-iso8859-1
courBO24.pcf	-adobe-courier-bold-o-normal--24-240-75-75-m-150-iso8859-1

Table B-3. Fonts in the 75dpi Directory (continued)

Filename	Font name
courBO08.pcf	-adobe-courier-bold-o-normal--8-80-75-75-m-50-iso8859-1
courB10.pcf	-adobe-courier-bold-r-normal--10-100-75-75-m-60-iso8859-1
courB12.pcf	-adobe-courier-bold-r-normal--12-120-75-75-m-70-iso8859-1
courB14.pcf	-adobe-courier-bold-r-normal--14-140-75-75-m-90-iso8859-1
courB18.pcf	-adobe-courier-bold-r-normal--18-180-75-75-m-110-iso8859-1
courB24.pcf	-adobe-courier-bold-r-normal--24-240-75-75-m-150-iso8859-1
courB08.pcf	-adobe-courier-bold-r-normal--8-80-75-75-m-50-iso8859-1
courO10.pcf	-adobe-courier-medium-o-normal--10-100-75-75-m-60-iso8859-1
courO12.pcf	-adobe-courier-medium-o-normal--12-120-75-75-m-70-iso8859-1
courO14.pcf	-adobe-courier-medium-o-normal--14-140-75-75-m-90-iso8859-1
courO18.pcf	-adobe-courier-medium-o-normal--18-180-75-75-m-110-iso8859-1
courO24.pcf	-adobe-courier-medium-o-normal--24-240-75-75-m-150-iso8859-1
courO08.pcf	-adobe-courier-medium-o-normal--8-80-75-75-m-50-iso8859-1
courR10.pcf	-adobe-courier-medium-r-normal--10-100-75-75-m-60-iso8859-1
courR12.pcf	-adobe-courier-medium-r-normal--12-120-75-75-m-70-iso8859-1
courR14.pcf	-adobe-courier-medium-r-normal--14-140-75-75-m-90-iso8859-1
courR18.pcf	-adobe-courier-medium-r-normal--18-180-75-75-m-110-iso8859-1
courR24.pcf	-adobe-courier-medium-r-normal--24-240-75-75-m-150-iso8859-1
courR08.pcf	-adobe-courier-medium-r-normal--8-80-75-75-m-50-iso8859-1
helvBO10.pcf	-adobe-helvetica-bold-o-normal--10-100-75-75-p-60-iso8859-1
helvBO12.pcf	-adobe-helvetica-bold-o-normal--12-120-75-75-p-69-iso8859-1
helvBO14.pcf	-adobe-helvetica-bold-o-normal--14-140-75-75-p-82-iso8859-1
helvBO18.pcf	-adobe-helvetica-bold-o-normal--18-180-75-75-p-104-iso8859-1
helvBO24.pcf	-adobe-helvetica-bold-o-normal--24-240-75-75-p-138-iso8859-1
helvBO08.pcf	-adobe-helvetica-bold-o-normal--8-80-75-75-p-50-iso8859-1
helvB10.pcf	-adobe-helvetica-bold-r-normal--10-100-75-75-p-60-iso8859-1
helvB12.pcf	-adobe-helvetica-bold-r-normal--12-120-75-75-p-70-iso8859-1
helvB14.pcf	-adobe-helvetica-bold-r-normal--14-140-75-75-p-82-iso8859-1
helvB18.pcf	-adobe-helvetica-bold-r-normal--18-180-75-75-p-103-iso8859-1
helvB24.pcf	-adobe-helvetica-bold-r-normal--24-240-75-75-p-138-iso8859-1
helvB08.pcf	-adobe-helvetica-bold-r-normal--8-80-75-75-p-50-iso8859-1
helvO10.pcf	-adobe-helvetica-medium-o-normal--10-100-75-75-p-57-iso8859-1
helvO12.pcf	-adobe-helvetica-medium-o-normal--12-120-75-75-p-67-iso8859-1
helvO14.pcf	-adobe-helvetica-medium-o-normal--14-140-75-75-p-78-iso8859-1
helvO18.pcf	-adobe-helvetica-medium-o-normal--18-180-75-75-p-98-iso8859-1
helvO24.pcf	-adobe-helvetica-medium-o-normal--24-240-75-75-p-130-iso8859-1
helvO08.pcf	-adobe-helvetica-medium-o-normal--8-80-75-75-p-47-iso8859-1
helvR10.pcf	-adobe-helvetica-medium-r-normal--10-100-75-75-p-56-iso8859-1
helvR12.pcf	-adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1
helvR14.pcf	-adobe-helvetica-medium-r-normal--14-140-75-75-p-77-iso8859-1
helvR18.pcf	-adobe-helvetica-medium-r-normal--18-180-75-75-p-98-iso8859-1
helvR24.pcf	-adobe-helvetica-medium-r-normal--24-240-75-75-p-130-iso8859-1
helvR08.pcf	-adobe-helvetica-medium-r-normal--8-80-75-75-p-46-iso8859-1

Table B-3. Fonts in the 75dpi Directory (continued)

Filename	Font name
ncenBI10.pcf	-adobe-new century schoolbook-bold-i-normal--10-100-75-75-p-66-iso8859-1
ncenBI12.pcf	-adobe-new century schoolbook-bold-i-normal--12-120-75-75-p-76-iso8859-1
ncenBI14.pcf	-adobe-new century schoolbook-bold-i-normal--14-140-75-75-p-88-iso8859-1
ncenBI18.pcf	-adobe-new century schoolbook-bold-i-normal--18-180-75-75-p-111-iso8859-1
ncenBI24.pcf	-adobe-new century schoolbook-bold-i-normal--24-240-75-75-p-148-iso8859-1
ncenBI08.pcf	-adobe-new century schoolbook-bold-i-normal--8-80-75-75-p-56-iso8859-1
ncenB10.pcf	-adobe-new century schoolbook-bold-r-normal--10-100-75-75-p-66-iso8859-1
ncenB12.pcf	-adobe-new century schoolbook-bold-r-normal--12-120-75-75-p-77-iso8859-1
ncenB14.pcf	-adobe-new century schoolbook-bold-r-normal--14-140-75-75-p-87-iso8859-1
ncenB18.pcf	-adobe-new century schoolbook-bold-r-normal--18-180-75-75-p-113-iso8859-1
ncenB24.pcf	-adobe-new century schoolbook-bold-r-normal--24-240-75-75-p-149-iso8859-1
ncenB08.pcf	-adobe-new century schoolbook-bold-r-normal--8-80-75-75-p-56-iso8859-1
ncenI10.pcf	-adobe-new century schoolbook-medium-i-normal--10-100-75-75-p-60-iso8859-1
ncenI12.pcf	-adobe-new century schoolbook-medium-i-normal--12-120-75-75-p-70-iso8859-1
ncenI14.pcf	-adobe-new century schoolbook-medium-i-normal--14-140-75-75-p-81-iso8859-1
ncenI18.pcf	-adobe-new century schoolbook-medium-i-normal--18-180-75-75-p-104-iso8859-1
ncenI24.pcf	-adobe-new century schoolbook-medium-i-normal--24-240-75-75-p-136-iso8859-1
ncenI08.pcf	-adobe-new century schoolbook-medium-i-normal--8-80-75-75-p-50-iso8859-1
ncenR10.pcf	-adobe-new century schoolbook-medium-r-normal--10-100-75-75-p-60-iso8859-1
ncenR12.pcf	-adobe-new century schoolbook-medium-r-normal--12-120-75-75-p-70-iso8859-1
ncenR14.pcf	-adobe-new century schoolbook-medium-r-normal--14-140-75-75-p-82-iso8859-1
ncenR18.pcf	-adobe-new century schoolbook-medium-r-normal--18-180-75-75-p-103-iso8859-1
ncenR24.pcf	-adobe-new century schoolbook-medium-r-normal--24-240-75-75-p-137-iso8859-1
ncenR08.pcf	-adobe-new century schoolbook-medium-r-normal--8-80-75-75-p-50-iso8859-1
symb10.pcf	-adobe-symbol-medium-r-normal--10-100-75-75-p-61-adobe-fontspecific
symb12.pcf	-adobe-symbol-medium-r-normal--12-120-75-75-p-74-adobe-fontspecific
symb14.pcf	-adobe-symbol-medium-r-normal--14-140-75-75-p-85-adobe-fontspecific
symb18.pcf	-adobe-symbol-medium-r-normal--18-180-75-75-p-107-adobe-fontspecific
symb24.pcf	-adobe-symbol-medium-r-normal--24-240-75-75-p-142-adobe-fontspecific
symb08.pcf	-adobe-symbol-medium-r-normal--8-80-75-75-p-51-adobe-fontspecific
timBI10.pcf	-adobe-times-bold-i-normal--10-100-75-75-p-57-iso8859-1
timBI12.pcf	-adobe-times-bold-i-normal--12-120-75-75-p-68-iso8859-1
timBI14.pcf	-adobe-times-bold-i-normal--14-140-75-75-p-77-iso8859-1
timBI18.pcf	-adobe-times-bold-i-normal--18-180-75-75-p-98-iso8859-1
timBI24.pcf	-adobe-times-bold-i-normal--24-240-75-75-p-128-iso8859-1
timBI08.pcf	-adobe-times-bold-i-normal--8-80-75-75-p-47-iso8859-1
timB10.pcf	-adobe-times-bold-r-normal--10-100-75-75-p-57-iso8859-1
timB12.pcf	-adobe-times-bold-r-normal--12-120-75-75-p-67-iso8859-1
timB14.pcf	-adobe-times-bold-r-normal--14-140-75-75-p-77-iso8859-1
timB18.pcf	-adobe-times-bold-r-normal--18-180-75-75-p-99-iso8859-1
timB24.pcf	-adobe-times-bold-r-normal--24-240-75-75-p-132-iso8859-1
timB08.pcf	-adobe-times-bold-r-normal--8-80-75-75-p-47-iso8859-1
timI10.pcf	-adobe-times-medium-i-normal--10-100-75-75-p-52-iso8859-1



Table B-3. Fonts in the 75dpi Directory (continued)

Filename	Font name
timI12.pcf	-adobe-times-medium-i-normal--12-120-75-75-p-63-iso8859-1
timI14.pcf	-adobe-times-medium-i-normal--14-140-75-75-p-73-iso8859-1
timI18.pcf	-adobe-times-medium-i-normal--18-180-75-75-p-94-iso8859-1
timI24.pcf	-adobe-times-medium-i-normal--24-240-75-75-p-125-iso8859-1
timI08.pcf	-adobe-times-medium-i-normal--8-80-75-75-p-42-iso8859-1
timR10.pcf	-adobe-times-medium-r-normal--10-100-75-75-p-54-iso8859-1
timR12.pcf	-adobe-times-medium-r-normal--12-120-75-75-p-64-iso8859-1
timR14.pcf	-adobe-times-medium-r-normal--14-140-75-75-p-74-iso8859-1
timR18.pcf	-adobe-times-medium-r-normal--18-180-75-75-p-94-iso8859-1
timR24.pcf	-adobe-times-medium-r-normal--24-240-75-75-p-124-iso8859-1
timR08.pcf	-adobe-times-medium-r-normal--8-80-75-75-p-44-iso8859-1
luBIS10.pcf	-b&h-lucida-bold-i-normal-sans-10-100-75-75-p-67-iso8859-1
luBIS12.pcf	-b&h-lucida-bold-i-normal-sans-12-120-75-75-p-79-iso8859-1
luBIS14.pcf	-b&h-lucida-bold-i-normal-sans-14-140-75-75-p-92-iso8859-1
luBIS18.pcf	-b&h-lucida-bold-i-normal-sans-18-180-75-75-p-119-iso8859-1
luBIS19.pcf	-b&h-lucida-bold-i-normal-sans-19-190-75-75-p-122-iso8859-1
luBIS24.pcf	-b&h-lucida-bold-i-normal-sans-24-240-75-75-p-151-iso8859-1
luBIS08.pcf	-b&h-lucida-bold-i-normal-sans-8-80-75-75-p-49-iso8859-1
luBS10.pcf	-b&h-lucida-bold-r-normal-sans-10-100-75-75-p-66-iso8859-1
luBS12.pcf	-b&h-lucida-bold-r-normal-sans-12-120-75-75-p-79-iso8859-1
luBS14.pcf	-b&h-lucida-bold-r-normal-sans-14-140-75-75-p-92-iso8859-1
luBS18.pcf	-b&h-lucida-bold-r-normal-sans-18-180-75-75-p-120-iso8859-1
luBS19.pcf	-b&h-lucida-bold-r-normal-sans-19-190-75-75-p-122-iso8859-1
luBS24.pcf	-b&h-lucida-bold-r-normal-sans-24-240-75-75-p-152-iso8859-1
luBS08.pcf	-b&h-lucida-bold-r-normal-sans-8-80-75-75-p-50-iso8859-1
luIS10.pcf	-b&h-lucida-medium-i-normal-sans-10-100-75-75-p-59-iso8859-1
luIS12.pcf	-b&h-lucida-medium-i-normal-sans-12-120-75-75-p-71-iso8859-1
luIS14.pcf	-b&h-lucida-medium-i-normal-sans-14-140-75-75-p-82-iso8859-1
luIS18.pcf	-b&h-lucida-medium-i-normal-sans-18-180-75-75-p-105-iso8859-1
luIS19.pcf	-b&h-lucida-medium-i-normal-sans-19-190-75-75-p-108-iso8859-1
luIS24.pcf	-b&h-lucida-medium-i-normal-sans-24-240-75-75-p-136-iso8859-1
luIS08.pcf	-b&h-lucida-medium-i-normal-sans-8-80-75-75-p-45-iso8859-1
luRS10.pcf	-b&h-lucida-medium-r-normal-sans-10-100-75-75-p-58-iso8859-1
luRS12.pcf	-b&h-lucida-medium-r-normal-sans-12-120-75-75-p-71-iso8859-1
luRS14.pcf	-b&h-lucida-medium-r-normal-sans-14-140-75-75-p-81-iso8859-1
luRS18.pcf	-b&h-lucida-medium-r-normal-sans-18-180-75-75-p-106-iso8859-1
luRS19.pcf	-b&h-lucida-medium-r-normal-sans-19-190-75-75-p-108-iso8859-1
luRS24.pcf	-b&h-lucida-medium-r-normal-sans-24-240-75-75-p-136-iso8859-1
luRS08.pcf	-b&h-lucida-medium-r-normal-sans-8-80-75-75-p-45-iso8859-1
lubBI10.pcf	-b&h-lucidabright-demibold-i-normal--10-100-75-75-p-59-iso8859-1
lubBI12.pcf	-b&h-lucidabright-demibold-i-normal--12-120-75-75-p-72-iso8859-1
lubBI14.pcf	-b&h-lucidabright-demibold-i-normal--14-140-75-75-p-84-iso8859-1
lubBI18.pcf	-b&h-lucidabright-demibold-i-normal--18-180-75-75-p-107-iso8859-1

Table B-3. Fonts in the 75dpi Directory (continued)

Filename	Font name
lubBI19.pcf	-b&h-lucidabright-demibold-i-normal--19-190-75-75-p-114-iso8859-1
lubBI24.pcf	-b&h-lucidabright-demibold-i-normal--24-240-75-75-p-143-iso8859-1
lubBI08.pcf	-b&h-lucidabright-demibold-i-normal--8-80-75-75-p-48-iso8859-1
lubB10.pcf	-b&h-lucidabright-demibold-r-normal--10-100-75-75-p-59-iso8859-1
lubB12.pcf	-b&h-lucidabright-demibold-r-normal--12-120-75-75-p-71-iso8859-1
lubB14.pcf	-b&h-lucidabright-demibold-r-normal--14-140-75-75-p-84-iso8859-1
lubB18.pcf	-b&h-lucidabright-demibold-r-normal--18-180-75-75-p-107-iso8859-1
lubB19.pcf	-b&h-lucidabright-demibold-r-normal--19-190-75-75-p-114-iso8859-1
lubB24.pcf	-b&h-lucidabright-demibold-r-normal--24-240-75-75-p-143-iso8859-1
lubB08.pcf	-b&h-lucidabright-demibold-r-normal--8-80-75-75-p-47-iso8859-1
lubI10.pcf	-b&h-lucidabright-medium-i-normal--10-100-75-75-p-57-iso8859-1
lubI12.pcf	-b&h-lucidabright-medium-i-normal--12-120-75-75-p-67-iso8859-1
lubI14.pcf	-b&h-lucidabright-medium-i-normal--14-140-75-75-p-80-iso8859-1
lubI18.pcf	-b&h-lucidabright-medium-i-normal--18-180-75-75-p-102-iso8859-1
lubI19.pcf	-b&h-lucidabright-medium-i-normal--19-190-75-75-p-109-iso8859-1
lubI24.pcf	-b&h-lucidabright-medium-i-normal--24-240-75-75-p-136-iso8859-1
lubI08.pcf	-b&h-lucidabright-medium-i-normal--8-80-75-75-p-45-iso8859-1
lubR10.pcf	-b&h-lucidabright-medium-r-normal--10-100-75-75-p-56-iso8859-1
lubR12.pcf	-b&h-lucidabright-medium-r-normal--12-120-75-75-p-68-iso8859-1
lubR14.pcf	-b&h-lucidabright-medium-r-normal--14-140-75-75-p-80-iso8859-1
lubR18.pcf	-b&h-lucidabright-medium-r-normal--18-180-75-75-p-103-iso8859-1
lubR19.pcf	-b&h-lucidabright-medium-r-normal--19-190-75-75-p-109-iso8859-1
lubR24.pcf	-b&h-lucidabright-medium-r-normal--24-240-75-75-p-137-iso8859-1
lubR08.pcf	-b&h-lucidabright-medium-r-normal--8-80-75-75-p-45-iso8859-1
lutBS10.pcf	-b&h-lucidatypewriter-bold-r-normal-sans-10-100-75-75-m-60-iso8859-1
lutBS12.pcf	-b&h-lucidatypewriter-bold-r-normal-sans-12-120-75-75-m-70-iso8859-1
lutBS14.pcf	-b&h-lucidatypewriter-bold-r-normal-sans-14-140-75-75-m-90-iso8859-1
lutBS18.pcf	-b&h-lucidatypewriter-bold-r-normal-sans-18-180-75-75-m-110-iso8859-1
lutBS19.pcf	-b&h-lucidatypewriter-bold-r-normal-sans-19-190-75-75-m-110-iso8859-1
lutBS24.pcf	-b&h-lucidatypewriter-bold-r-normal-sans-24-240-75-75-m-140-iso8859-1
lutBS08.pcf	-b&h-lucidatypewriter-bold-r-normal-sans-8-80-75-75-m-50-iso8859-1
lutRS10.pcf	-b&h-lucidatypewriter-medium-r-normal-sans-10-100-75-75-m-60-iso8859-1
lutRS12.pcf	-b&h-lucidatypewriter-medium-r-normal-sans-12-120-75-75-m-70-iso8859-1
lutRS14.pcf	-b&h-lucidatypewriter-medium-r-normal-sans-14-140-75-75-m-90-iso8859-1
lutRS18.pcf	-b&h-lucidatypewriter-medium-r-normal-sans-18-180-75-75-m-110-iso8859-1
lutRS19.pcf	-b&h-lucidatypewriter-medium-r-normal-sans-19-190-75-75-m-110-iso8859-1
lutRS24.pcf	-b&h-lucidatypewriter-medium-r-normal-sans-24-240-75-75-m-140-iso8859-1
lutRS08.pcf	-b&h-lucidatypewriter-medium-r-normal-sans-8-80-75-75-m-50-iso8859-1
charBI10.pcf	-bitstream-charter-bold-i-normal--10-100-75-75-p-62-iso8859-1
charBI12.pcf	-bitstream-charter-bold-i-normal--12-120-75-75-p-74-iso8859-1
charBI14.pcf	-bitstream-charter-bold-i-normal--15-140-75-75-p-93-iso8859-1
charBI18.pcf	-bitstream-charter-bold-i-normal--19-180-75-75-p-117-iso8859-1
charBI24.pcf	-bitstream-charter-bold-i-normal--25-240-75-75-p-154-iso8859-1

Release 5
Standard Fonts

Table B-3. Fonts in the 75dpi Directory (continued)

Filename	Font name
charBI08.pcf	-bitstream-charter-bold-i-normal--8-80-75-75-p-50-iso8859-1
charB10.pcf	-bitstream-charter-bold-r-normal--10-100-75-75-p-63-iso8859-1
charB12.pcf	-bitstream-charter-bold-r-normal--12-120-75-75-p-75-iso8859-1
charB14.pcf	-bitstream-charter-bold-r-normal--15-140-75-75-p-94-iso8859-1
charB18.pcf	-bitstream-charter-bold-r-normal--19-180-75-75-p-119-iso8859-1
charB24.pcf	-bitstream-charter-bold-r-normal--25-240-75-75-p-157-iso8859-1
charB08.pcf	-bitstream-charter-bold-r-normal--8-80-75-75-p-50-iso8859-1
charI10.pcf	-bitstream-charter-medium-i-normal--10-100-75-75-p-55-iso8859-1
charI12.pcf	-bitstream-charter-medium-i-normal--12-120-75-75-p-65-iso8859-1
charI14.pcf	-bitstream-charter-medium-i-normal--15-140-75-75-p-82-iso8859-1
charI18.pcf	-bitstream-charter-medium-i-normal--19-180-75-75-p-103-iso8859-1
charI24.pcf	-bitstream-charter-medium-i-normal--25-240-75-75-p-136-iso8859-1
charI08.pcf	-bitstream-charter-medium-i-normal--8-80-75-75-p-44-iso8859-1
charR10.pcf	-bitstream-charter-medium-r-normal--10-100-75-75-p-56-iso8859-1
charR12.pcf	-bitstream-charter-medium-r-normal--12-120-75-75-p-67-iso8859-1
charR14.pcf	-bitstream-charter-medium-r-normal--15-140-75-75-p-84-iso8859-1
charR18.pcf	-bitstream-charter-medium-r-normal--19-180-75-75-p-106-iso8859-1
charR24.pcf	-bitstream-charter-medium-r-normal--25-240-75-75-p-139-iso8859-1
charR08.pcf	-bitstream-charter-medium-r-normal--8-80-75-75-p-45-iso8859-1
techB14.pcf	-dec-terminal-bold-r-normal--14-140-75-75-c-80-dec-dectech
termB14.pcf	-dec-terminal-bold-r-normal--14-140-75-75-c-80-iso8859-1
tech14.pcf	-dec-terminal-medium-r-normal--14-140-75-75-c-80-dec-dectech
term14.pcf	-dec-terminal-medium-r-normal--14-140-75-75-c-80-iso8859-1

Table B-4. Fonts in the 100dpi Directory

Filename	Font name
courBO08.pcf	-adobe-courier-bold-o-normal--11-80-100-100-m-60-iso8859-1
courBO10.pcf	-adobe-courier-bold-o-normal--14-100-100-100-m-90-iso8859-1
courBO12.pcf	-adobe-courier-bold-o-normal--17-120-100-100-m-100-iso8859-1
courBO14.pcf	-adobe-courier-bold-o-normal--20-140-100-100-m-110-iso8859-1
courBO18.pcf	-adobe-courier-bold-o-normal--25-180-100-100-m-150-iso8859-1
courBO24.pcf	-adobe-courier-bold-o-normal--34-240-100-100-m-200-iso8859-1
courB08.pcf	-adobe-courier-bold-r-normal--11-80-100-100-m-60-iso8859-1
courB10.pcf	-adobe-courier-bold-r-normal--14-100-100-100-m-90-iso8859-1
courB12.pcf	-adobe-courier-bold-r-normal--17-120-100-100-m-100-iso8859-1
courB14.pcf	-adobe-courier-bold-r-normal--20-140-100-100-m-110-iso8859-1
courB18.pcf	-adobe-courier-bold-r-normal--25-180-100-100-m-150-iso8859-1
courB24.pcf	-adobe-courier-bold-r-normal--34-240-100-100-m-200-iso8859-1
courO08.pcf	-adobe-courier-medium-o-normal--11-80-100-100-m-60-iso8859-1
courO10.pcf	-adobe-courier-medium-o-normal--14-100-100-100-m-90-iso8859-1

Table B-4. Fonts in the 100dpi Directory (continued)

Filename	Font name
courO12.pcf	-adobe-courier-medium-o-normal--17-120-100-100-m-100-iso8859-1
courO14.pcf	-adobe-courier-medium-o-normal--20-140-100-100-m-110-iso8859-1
courO18.pcf	-adobe-courier-medium-o-normal--25-180-100-100-m-150-iso8859-1
courO24.pcf	-adobe-courier-medium-o-normal--34-240-100-100-m-200-iso8859-1
courR08.pcf	-adobe-courier-medium-r-normal--11-80-100-100-m-60-iso8859-1
courR10.pcf	-adobe-courier-medium-r-normal--14-100-100-100-m-90-iso8859-1
courR12.pcf	-adobe-courier-medium-r-normal--17-120-100-100-m-100-iso8859-1
courR14.pcf	-adobe-courier-medium-r-normal--20-140-100-100-m-110-iso8859-1
courR18.pcf	-adobe-courier-medium-r-normal--25-180-100-100-m-150-iso8859-1
courR24.pcf	-adobe-courier-medium-r-normal--34-240-100-100-m-200-iso8859-1
helvBO08.pcf	-adobe-helvetica-bold-o-normal--11-80-100-100-p-60-iso8859-1
helvBO10.pcf	-adobe-helvetica-bold-o-normal--14-100-100-100-p-82-iso8859-1
helvBO12.pcf	-adobe-helvetica-bold-o-normal--17-120-100-100-p-92-iso8859-1
helvBO14.pcf	-adobe-helvetica-bold-o-normal--20-140-100-100-p-103-iso8859-1
helvBO18.pcf	-adobe-helvetica-bold-o-normal--25-180-100-100-p-138-iso8859-1
helvBO24.pcf	-adobe-helvetica-bold-o-normal--34-240-100-100-p-182-iso8859-1
helvB08.pcf	-adobe-helvetica-bold-r-normal--11-80-100-100-p-60-iso8859-1
helvB10.pcf	-adobe-helvetica-bold-r-normal--14-100-100-100-p-82-iso8859-1
helvB12.pcf	-adobe-helvetica-bold-r-normal--17-120-100-100-p-92-iso8859-1
helvB14.pcf	-adobe-helvetica-bold-r-normal--20-140-100-100-p-105-iso8859-1
helvB18.pcf	-adobe-helvetica-bold-r-normal--25-180-100-100-p-138-iso8859-1
helvB24.pcf	-adobe-helvetica-bold-r-normal--34-240-100-100-p-182-iso8859-1
helvO08.pcf	-adobe-helvetica-medium-o-normal--11-80-100-100-p-57-iso8859-1
helvO10.pcf	-adobe-helvetica-medium-o-normal--14-100-100-100-p-78-iso8859-1
helvO12.pcf	-adobe-helvetica-medium-o-normal--17-120-100-100-p-88-iso8859-1
helvO14.pcf	-adobe-helvetica-medium-o-normal--20-140-100-100-p-98-iso8859-1
helvO18.pcf	-adobe-helvetica-medium-o-normal--25-180-100-100-p-130-iso8859-1
helvO24.pcf	-adobe-helvetica-medium-o-normal--34-240-100-100-p-176-iso8859-1
helvR08.pcf	-adobe-helvetica-medium-r-normal--11-80-100-100-p-56-iso8859-1
helvR10.pcf	-adobe-helvetica-medium-r-normal--14-100-100-100-p-76-iso8859-1
helvR12.pcf	-adobe-helvetica-medium-r-normal--17-120-100-100-p-88-iso8859-1
helvR14.pcf	-adobe-helvetica-medium-r-normal--20-140-100-100-p-100-iso8859-1
helvR18.pcf	-adobe-helvetica-medium-r-normal--25-180-100-100-p-130-iso8859-1
helvR24.pcf	-adobe-helvetica-medium-r-normal--34-240-100-100-p-176-iso8859-1
ncenBI08.pcf	-adobe-new century schoolbook-bold-i-normal--11-80-100-100-p-66-iso8859-1
ncenBI10.pcf	-adobe-new century schoolbook-bold-i-normal--14-100-100-100-p-88-iso8859-1
ncenBI12.pcf	-adobe-new century schoolbook-bold-i-normal--17-120-100-100-p-99-iso8859-1
ncenBI14.pcf	-adobe-new century schoolbook-bold-i-normal--20-140-100-100-p-111-iso8859-1
ncenBI18.pcf	-adobe-new century schoolbook-bold-i-normal--25-180-100-100-p-148-iso8859-1
ncenBI24.pcf	-adobe-new century schoolbook-bold-i-normal--34-240-100-100-p-193-iso8859-1
ncenB08.pcf	-adobe-new century schoolbook-bold-r-normal--11-80-100-100-p-66-iso8859-1
ncenB10.pcf	-adobe-new century schoolbook-bold-r-normal--14-100-100-100-p-87-iso8859-1
ncenB12.pcf	-adobe-new century schoolbook-bold-r-normal--17-120-100-100-p-99-iso8859-1
ncenB14.pcf	-adobe-new century schoolbook-bold-r-normal--20-140-100-100-p-113-iso8859-1

Release 5
Standard Fonts

Table B-4. Fonts in the 100dpi Directory (continued)

Filename	Font name
ncenB18.pcf	-adobe-new century schoolbook-bold-r-normal--25-180-100-100-p-149-iso8859-1
ncenB24.pcf	-adobe-new century schoolbook-bold-r-normal--34-240-100-100-p-193-iso8859-1
ncenI08.pcf	-adobe-new century schoolbook-medium-i-normal--11-80-100-100-p-60-iso8859-1
ncenI10.pcf	-adobe-new century schoolbook-medium-i-normal--14-100-100-100-p-81-iso8859-1
ncenI12.pcf	-adobe-new century schoolbook-medium-i-normal--17-120-100-100-p-92-iso8859-1
ncenI14.pcf	-adobe-new century schoolbook-medium-i-normal--20-140-100-100-p-104-iso8859-1
ncenI18.pcf	-adobe-new century schoolbook-medium-i-normal--25-180-100-100-p-136-iso8859-1
ncenI24.pcf	-adobe-new century schoolbook-medium-i-normal--34-240-100-100-p-182-iso8859-1
ncenR08.pcf	-adobe-new century schoolbook-medium-r-normal--11-80-100-100-p-60-iso8859-1
ncenR10.pcf	-adobe-new century schoolbook-medium-r-normal--14-100-100-100-p-82-iso8859-1
ncenR12.pcf	-adobe-new century schoolbook-medium-r-normal--17-120-100-100-p-91-iso8859-1
ncenR14.pcf	-adobe-new century schoolbook-medium-r-normal--20-140-100-100-p-103-iso8859-1
ncenR18.pcf	-adobe-new century schoolbook-medium-r-normal--25-180-100-100-p-136-iso8859-1
ncenR24.pcf	-adobe-new century schoolbook-medium-r-normal--34-240-100-100-p-181-iso8859-1
symb08.pcf	-adobe-symbol-medium-r-normal--11-80-100-100-p-61-adobe-fontspecific
symb10.pcf	-adobe-symbol-medium-r-normal--14-100-100-100-p-85-adobe-fontspecific
symb12.pcf	-adobe-symbol-medium-r-normal--17-120-100-100-p-95-adobe-fontspecific
symb14.pcf	-adobe-symbol-medium-r-normal--20-140-100-100-p-107-adobe-fontspecific
symb18.pcf	-adobe-symbol-medium-r-normal--25-180-100-100-p-142-adobe-fontspecific
symb24.pcf	-adobe-symbol-medium-r-normal--34-240-100-100-p-191-adobe-fontspecific
timBI08.pcf	-adobe-times-bold-i-normal--11-80-100-100-p-57-iso8859-1
timBI10.pcf	-adobe-times-bold-i-normal--14-100-100-100-p-77-iso8859-1
timBI12.pcf	-adobe-times-bold-i-normal--17-120-100-100-p-86-iso8859-1
timBI14.pcf	-adobe-times-bold-i-normal--20-140-100-100-p-98-iso8859-1
timBI18.pcf	-adobe-times-bold-i-normal--25-180-100-100-p-128-iso8859-1
timBI24.pcf	-adobe-times-bold-i-normal--34-240-100-100-p-170-iso8859-1
timB08.pcf	-adobe-times-bold-r-normal--11-80-100-100-p-57-iso8859-1
timB10.pcf	-adobe-times-bold-r-normal--14-100-100-100-p-76-iso8859-1
timB12.pcf	-adobe-times-bold-r-normal--17-120-100-100-p-88-iso8859-1
timB14.pcf	-adobe-times-bold-r-normal--20-140-100-100-p-100-iso8859-1
timB18.pcf	-adobe-times-bold-r-normal--25-180-100-100-p-132-iso8859-1
timB24.pcf	-adobe-times-bold-r-normal--34-240-100-100-p-177-iso8859-1
timI08.pcf	-adobe-times-medium-i-normal--11-80-100-100-p-52-iso8859-1
timI10.pcf	-adobe-times-medium-i-normal--14-100-100-100-p-73-iso8859-1
timI12.pcf	-adobe-times-medium-i-normal--17-120-100-100-p-84-iso8859-1
timI14.pcf	-adobe-times-medium-i-normal--20-140-100-100-p-94-iso8859-1
timI18.pcf	-adobe-times-medium-i-normal--25-180-100-100-p-125-iso8859-1
timI24.pcf	-adobe-times-medium-i-normal--34-240-100-100-p-168-iso8859-1
timR08.pcf	-adobe-times-medium-r-normal--11-80-100-100-p-54-iso8859-1
timR10.pcf	-adobe-times-medium-r-normal--14-100-100-100-p-74-iso8859-1
timR12.pcf	-adobe-times-medium-r-normal--17-120-100-100-p-84-iso8859-1
timR14.pcf	-adobe-times-medium-r-normal--20-140-100-100-p-96-iso8859-1
timR18.pcf	-adobe-times-medium-r-normal--25-180-100-100-p-125-iso8859-1
timR24.pcf	-adobe-times-medium-r-normal--34-240-100-100-p-170-iso8859-1

Table B-4. Fonts in the 100dpi Directory (continued)

Filename	Font name
luBIS08.pcf	-b&h-lucida-bold-i-normal-sans-11-80-100-100-p-69-iso8859-1
luBIS10.pcf	-b&h-lucida-bold-i-normal-sans-14-100-100-100-p-90-iso8859-1
luBIS12.pcf	-b&h-lucida-bold-i-normal-sans-17-120-100-100-p-108-iso8859-1
luBIS14.pcf	-b&h-lucida-bold-i-normal-sans-20-140-100-100-p-127-iso8859-1
luBIS18.pcf	-b&h-lucida-bold-i-normal-sans-25-180-100-100-p-159-iso8859-1
luBIS19.pcf	-b&h-lucida-bold-i-normal-sans-26-190-100-100-p-166-iso8859-1
luBIS24.pcf	-b&h-lucida-bold-i-normal-sans-34-240-100-100-p-215-iso8859-1
luBS08.pcf	-b&h-lucida-bold-r-normal-sans-11-80-100-100-p-70-iso8859-1
luBS10.pcf	-b&h-lucida-bold-r-normal-sans-14-100-100-100-p-89-iso8859-1
luBS12.pcf	-b&h-lucida-bold-r-normal-sans-17-120-100-100-p-108-iso8859-1
luBS14.pcf	-b&h-lucida-bold-r-normal-sans-20-140-100-100-p-127-iso8859-1
luBS18.pcf	-b&h-lucida-bold-r-normal-sans-25-180-100-100-p-158-iso8859-1
luBS19.pcf	-b&h-lucida-bold-r-normal-sans-26-190-100-100-p-166-iso8859-1
luBS24.pcf	-b&h-lucida-bold-r-normal-sans-34-240-100-100-p-216-iso8859-1
luIS08.pcf	-b&h-lucida-medium-i-normal-sans-11-80-100-100-p-62-iso8859-1
luIS10.pcf	-b&h-lucida-medium-i-normal-sans-14-100-100-100-p-80-iso8859-1
luIS12.pcf	-b&h-lucida-medium-i-normal-sans-17-120-100-100-p-97-iso8859-1
luIS14.pcf	-b&h-lucida-medium-i-normal-sans-20-140-100-100-p-114-iso8859-1
luIS18.pcf	-b&h-lucida-medium-i-normal-sans-25-180-100-100-p-141-iso8859-1
luIS19.pcf	-b&h-lucida-medium-i-normal-sans-26-190-100-100-p-147-iso8859-1
luIS24.pcf	-b&h-lucida-medium-i-normal-sans-34-240-100-100-p-192-iso8859-1
luRS08.pcf	-b&h-lucida-medium-r-normal-sans-11-80-100-100-p-63-iso8859-1
luRS10.pcf	-b&h-lucida-medium-r-normal-sans-14-100-100-100-p-80-iso8859-1
luRS12.pcf	-b&h-lucida-medium-r-normal-sans-17-120-100-100-p-96-iso8859-1
luRS14.pcf	-b&h-lucida-medium-r-normal-sans-20-140-100-100-p-114-iso8859-1
luRS18.pcf	-b&h-lucida-medium-r-normal-sans-25-180-100-100-p-142-iso8859-1
luRS19.pcf	-b&h-lucida-medium-r-normal-sans-26-190-100-100-p-147-iso8859-1
luRS24.pcf	-b&h-lucida-medium-r-normal-sans-34-240-100-100-p-191-iso8859-1
lubBI08.pcf	-b&h-lucidabright-demibold-i-normal--11-80-100-100-p-66-iso8859-1
lubBI10.pcf	-b&h-lucidabright-demibold-i-normal--14-100-100-100-p-84-iso8859-1
lubBI12.pcf	-b&h-lucidabright-demibold-i-normal--17-120-100-100-p-101-iso8859-1
lubBI14.pcf	-b&h-lucidabright-demibold-i-normal--20-140-100-100-p-119-iso8859-1
lubBI18.pcf	-b&h-lucidabright-demibold-i-normal--25-180-100-100-p-149-iso8859-1
lubBI19.pcf	-b&h-lucidabright-demibold-i-normal--26-190-100-100-p-156-iso8859-1
lubBI24.pcf	-b&h-lucidabright-demibold-i-normal--34-240-100-100-p-203-iso8859-1
lubB08.pcf	-b&h-lucidabright-demibold-r-normal--11-80-100-100-p-66-iso8859-1
lubB10.pcf	-b&h-lucidabright-demibold-r-normal--14-100-100-100-p-84-iso8859-1
lubB12.pcf	-b&h-lucidabright-demibold-r-normal--17-120-100-100-p-101-iso8859-1
lubB14.pcf	-b&h-lucidabright-demibold-r-normal--20-140-100-100-p-118-iso8859-1
lubB18.pcf	-b&h-lucidabright-demibold-r-normal--25-180-100-100-p-149-iso8859-1
lubB19.pcf	-b&h-lucidabright-demibold-r-normal--26-190-100-100-p-155-iso8859-1
lubB24.pcf	-b&h-lucidabright-demibold-r-normal--34-240-100-100-p-202-iso8859-1
lubI08.pcf	-b&h-lucidabright-medium-i-normal--11-80-100-100-p-63-iso8859-1
lubI10.pcf	-b&h-lucidabright-medium-i-normal--14-100-100-100-p-80-iso8859-1

Release 5
Standard Fonts

Table B-4. Fonts in the 100dpi Directory (continued)

Filename	Font name
lubI12.pcf	-b&h-lucidabright-medium-i-normal--17-120-100-100-p-96-iso8859-1
lubI14.pcf	-b&h-lucidabright-medium-i-normal--20-140-100-100-p-113-iso8859-1
lubI18.pcf	-b&h-lucidabright-medium-i-normal--25-180-100-100-p-142-iso8859-1
lubI19.pcf	-b&h-lucidabright-medium-i-normal--26-190-100-100-p-148-iso8859-1
lubI24.pcf	-b&h-lucidabright-medium-i-normal--34-240-100-100-p-194-iso8859-1
lubR08.pcf	-b&h-lucidabright-medium-r-normal--11-80-100-100-p-63-iso8859-1
lubR10.pcf	-b&h-lucidabright-medium-r-normal--14-100-100-100-p-80-iso8859-1
lubR12.pcf	-b&h-lucidabright-medium-r-normal--17-120-100-100-p-96-iso8859-1
lubR14.pcf	-b&h-lucidabright-medium-r-normal--20-140-100-100-p-114-iso8859-1
lubR18.pcf	-b&h-lucidabright-medium-r-normal--25-180-100-100-p-142-iso8859-1
lubR19.pcf	-b&h-lucidabright-medium-r-normal--26-190-100-100-p-149-iso8859-1
lubR24.pcf	-b&h-lucidabright-medium-r-normal--34-240-100-100-p-193-iso8859-1
lutBS08.pcf	-b&h-lucidatypewriter-bold-r-normal-sans-11-80-100-100-m-70-iso8859-1
lutBS10.pcf	-b&h-lucidatypewriter-bold-r-normal-sans-14-100-100-100-m-80-iso8859-1
lutBS12.pcf	-b&h-lucidatypewriter-bold-r-normal-sans-17-120-100-100-m-100-iso8859-1
lutBS14.pcf	-b&h-lucidatypewriter-bold-r-normal-sans-20-140-100-100-m-120-iso8859-1
lutBS18.pcf	-b&h-lucidatypewriter-bold-r-normal-sans-25-180-100-100-m-150-iso8859-1
lutBS19.pcf	-b&h-lucidatypewriter-bold-r-normal-sans-26-190-100-100-m-159-iso8859-1
lutBS24.pcf	-b&h-lucidatypewriter-bold-r-normal-sans-34-240-100-100-m-200-iso8859-1
lutRS08.pcf	-b&h-lucidatypewriter-medium-r-normal-sans-11-80-100-100-m-70-iso8859-1
lutRS10.pcf	-b&h-lucidatypewriter-medium-r-normal-sans-14-100-100-100-m-80-iso8859-1
lutRS12.pcf	-b&h-lucidatypewriter-medium-r-normal-sans-17-120-100-100-m-100-iso8859-1
lutRS14.pcf	-b&h-lucidatypewriter-medium-r-normal-sans-20-140-100-100-m-120-iso8859-1
lutRS18.pcf	-b&h-lucidatypewriter-medium-r-normal-sans-25-180-100-100-m-150-iso8859-1
lutRS19.pcf	-b&h-lucidatypewriter-medium-r-normal-sans-26-190-100-100-m-159-iso8859-1
lutRS24.pcf	-b&h-lucidatypewriter-medium-r-normal-sans-34-240-100-100-m-200-iso8859-1
charBI08.pcf	-bitstream-charter-bold-i-normal--11-80-100-100-p-68-iso8859-1
charBI10.pcf	-bitstream-charter-bold-i-normal--14-100-100-100-p-86-iso8859-1
charBI12.pcf	-bitstream-charter-bold-i-normal--17-120-100-100-p-105-iso8859-1
charBI14.pcf	-bitstream-charter-bold-i-normal--19-140-100-100-p-117-iso8859-1
charBI18.pcf	-bitstream-charter-bold-i-normal--25-180-100-100-p-154-iso8859-1
charBI24.pcf	-bitstream-charter-bold-i-normal--33-240-100-100-p-203-iso8859-1
charB08.pcf	-bitstream-charter-bold-r-normal--11-80-100-100-p-69-iso8859-1
charB10.pcf	-bitstream-charter-bold-r-normal--14-100-100-100-p-88-iso8859-1
charB12.pcf	-bitstream-charter-bold-r-normal--17-120-100-100-p-107-iso8859-1
charB14.pcf	-bitstream-charter-bold-r-normal--19-140-100-100-p-119-iso8859-1
charB18.pcf	-bitstream-charter-bold-r-normal--25-180-100-100-p-157-iso8859-1
charB24.pcf	-bitstream-charter-bold-r-normal--33-240-100-100-p-206-iso8859-1
charI08.pcf	-bitstream-charter-medium-i-normal--11-80-100-100-p-60-iso8859-1
charI10.pcf	-bitstream-charter-medium-i-normal--14-100-100-100-p-76-iso8859-1
charI12.pcf	-bitstream-charter-medium-i-normal--17-120-100-100-p-92-iso8859-1
charI14.pcf	-bitstream-charter-medium-i-normal--19-140-100-100-p-103-iso8859-1
charI18.pcf	-bitstream-charter-medium-i-normal--25-180-100-100-p-136-iso8859-1
charI24.pcf	-bitstream-charter-medium-i-normal--33-240-100-100-p-179-iso8859-1

Table B-4. Fonts in the 100dpi Directory (continued)

Filename	Font name
charR08.pcf	-bitstream-charter-medium-r-normal--11-80-100-100-p-61-iso8859-1
charR10.pcf	-bitstream-charter-medium-r-normal--14-100-100-100-p-78-iso8859-1
charR12.pcf	-bitstream-charter-medium-r-normal--17-120-100-100-p-95-iso8859-1
charR14.pcf	-bitstream-charter-medium-r-normal--19-140-100-100-p-106-iso8859-1
charR18.pcf	-bitstream-charter-medium-r-normal--25-180-100-100-p-139-iso8859-1
charR24.pcf	-bitstream-charter-medium-r-normal--33-240-100-100-p-183-iso8859-1
techB14.pcf	-bitstream-terminal-bold-r-normal--18-140-100-100-c-110-dec-dectech
termB14.pcf	-bitstream-terminal-bold-r-normal--18-140-100-100-c-110-iso8859-1
tech14.pcf	-bitstream-terminal-medium-r-normal--18-140-100-100-c-110-dec-dectech
term14.pcf	-bitstream-terminal-medium-r-normal--18-140-100-100-c-110-iso8859-1

Table B-5. Fonts in the Speedo directory

Filename	Font name
font0648.spd	-bitstream-charter-medium-r-normal--0-0-0-0-p-0-iso8859-1
font0649.spd	-bitstream-charter-medium-i-normal--0-0-0-0-p-0-iso8859-1
font0709.spd	-bitstream-charter-bold-r-normal--0-0-0-0-p-0-iso8859-1
font0710.spd	-bitstream-charter-bold-i-normal--0-0-0-0-p-0-iso8859-1
font0419.spd	-bitstream-courier-medium-r-normal--0-0-0-0-m-0-iso8859-1
font0582.spd	-bitstream-courier-medium-i-normal--0-0-0-0-m-0-iso8859-1
font0583.spd	-bitstream-courier-bold-r-normal--0-0-0-0-m-0-iso8859-1
font0611.spd	-bitstream-courier-bold-i-normal--0-0-0-0-m-0-iso8859-1

Release 5
Standard Fonts

Foundry: adobe
Family: courier

-adobe-courier-medium-o-normal--8-80-75-75-m-50-iso8859-1
-adobe-courier-medium-o-normal--10-100-75-75-m-60-iso8859-1
-adobe-courier-medium-o-normal--12-120-75-75-m-70-iso8859-1
-adobe-courier-medium-o-normal--14-140-75-75-m-90-iso8859-1
-adobe-courier-medium-o-normal--18-180-75-75-m-110-iso8859-1
-adobe-courier-medium-o-normal--24-240-75-75-m-1
-adobe-courier-medium-r-normal--8-80-75-75-m-50-iso8859-1
-adobe-courier-medium-r-normal--10-100-75-75-m-60-iso8859-1
-adobe-courier-medium-r-normal--12-120-75-75-m-70-iso8859-1
-adobe-courier-medium-r-normal--14-140-75-75-m-90-iso8859-1
-adobe-courier-medium-r-normal--18-180-75-75-m-110-iso8859-1
-adobe-courier-medium-r-normal--24-240-75-75-m-1
-adobe-courier-bold-o-normal--8-80-75-75-m-50-iso8859-1
-adobe-courier-bold-o-normal--10-100-75-75-m-60-iso8859-1
-adobe-courier-bold-o-normal--12-120-75-75-m-70-iso8859-1
-adobe-courier-bold-o-normal--14-140-75-75-m-90-iso8859-1
-adobe-courier-bold-o-normal--18-180-75-75-m-110-iso8859-1
-adobe-courier-bold-o-normal--24-240-75-75-m-1
-adobe-courier-bold-r-normal--8-80-75-75-m-50-iso8859-1
-adobe-courier-bold-r-normal--10-100-75-75-m-60-iso8859-1
-adobe-courier-bold-r-normal--12-120-75-75-m-70-iso8859-1
-adobe-courier-bold-r-normal--14-140-75-75-m-90-iso8859-1
-adobe-courier-bold-r-normal--18-180-75-75-m-110-iso8859-1
-adobe-courier-bold-r-normal--24-240-75-75-m-1

Foundry: adobe
Family: helvetica

- adobe-helvetica-medium-o-normal--8-80-75-75-p-47-iso8859-1
- adobe-helvetica-medium-o-normal--10-100-75-75-p-57-iso8859-1
- adobe-helvetica-medium-o-normal--12-120-75-75-p-67-iso8859-1
- adobe-helvetica-medium-o-normal--14-140-75-75-p-78-iso8859-1
- adobe-helvetica-medium-o-normal--18-180-75-75-p-98-iso8859-1**
- adobe-helvetica-medium-o-normal--24-240-75-75-p-130**
- adobe-helvetica-medium-r-normal--8-80-75-75-p-46-iso8859-1
- adobe-helvetica-medium-r-normal--10-100-75-75-p-56-iso8859-1
- adobe-helvetica-medium-r-normal--12-120-75-75-p-67-iso8859-1
- adobe-helvetica-medium-r-normal--14-140-75-75-p-77-iso8859-1
- adobe-helvetica-medium-r-normal--18-180-75-75-p-98-iso8859-1**
- adobe-helvetica-medium-r-normal--24-240-75-75-p-130**
- adobe-helvetica-bold-o-normal--8-80-75-75-p-50-iso8859-1
- adobe-helvetica-bold-o-normal--10-100-75-75-p-60-iso8859-1
- adobe-helvetica-bold-o-normal--12-120-75-75-p-69-iso8859-1
- adobe-helvetica-bold-o-normal--14-140-75-75-p-82-iso8859-1
- adobe-helvetica-bold-o-normal--18-180-75-75-p-104-iso8859-1**
- adobe-helvetica-bold-o-normal--24-240-75-75-p-138-**
- adobe-helvetica-bold-r-normal--8-80-75-75-p-50-iso8859-1
- adobe-helvetica-bold-r-normal--10-100-75-75-p-60-iso8859-1
- adobe-helvetica-bold-r-normal--12-120-75-75-p-70-iso8859-1
- adobe-helvetica-bold-r-normal--14-140-75-75-p-82-iso8859-1
- adobe-helvetica-bold-r-normal--18-180-75-75-p-103-iso8859-1**
- adobe-helvetica-bold-r-normal--24-240-75-75-p-138-**

Foundry: adobe
Family: new century schoolbook

-adobe-new century schoolbook-medium-i-normal--4-40-75-75-p-50-iso8859-1
-adobe-new century schoolbook-medium-i-normal--10-100-75-75-p-60-iso8859-1
-adobe-new century schoolbook-medium-i-normal--12-120-75-75-p-70-iso8859-1
-adobe-new century schoolbook-medium-i-normal--14-140-75-75-p-81-iso8859-1
-adobe-new century schoolbook-medium-i-normal--18-180-75-75-p-104-iso8859-1
-adobe-new century schoolbook-medium-i-normal--24-240-75-75-p-136-iso8859-1
-adobe-new century schoolbook-medium-r-normal--4-40-75-75-p-50-iso8859-1
-adobe-new century schoolbook-medium-r-normal--10-100-75-75-p-60-iso8859-1
-adobe-new century schoolbook-medium-r-normal--12-120-75-75-p-70-iso8859-1
-adobe-new century schoolbook-medium-r-normal--14-140-75-75-p-82-iso8859-1
-adobe-new century schoolbook-medium-r-normal--18-180-75-75-p-103-iso8859-1
-adobe-new century schoolbook-medium-r-normal--24-240-75-75-p-137-iso8859-1
-adobe-new century schoolbook-bold-i-normal--6-60-75-75-p-56-iso8859-1
-adobe-new century schoolbook-bold-i-normal--10-100-75-75-p-66-iso8859-1
-adobe-new century schoolbook-bold-i-normal--12-120-75-75-p-76-iso8859-1
-adobe-new century schoolbook-bold-i-normal--14-140-75-75-p-88-iso8859-1
-adobe-new century schoolbook-bold-i-normal--18-180-75-75-p-111-iso8859-1
-adobe-new century schoolbook-bold-i-normal--24-240-75-75-p-148-iso8859-1
-adobe-new century schoolbook-bold-r-normal--6-60-75-75-p-56-iso8859-1
-adobe-new century schoolbook-bold-r-normal--10-100-75-75-p-66-iso8859-1
-adobe-new century schoolbook-bold-r-normal--12-120-75-75-p-77-iso8859-1
-adobe-new century schoolbook-bold-r-normal--14-140-75-75-p-87-iso8859-1
-adobe-new century schoolbook-bold-r-normal--18-180-75-75-p-113-iso8859-1
-adobe-new century schoolbook-bold-r-normal--24-240-75-75-p-149-iso8859-1

Foundry: b&h
Family: lucida

- b&h-lucida-medium-i-normal-sans-8-80-75-75-p-45-iso8859-1
- b&h-lucida-medium-i-normal-sans-10-100-75-75-p-59-iso8859-1
- b&h-lucida-medium-i-normal-sans-12-120-75-75-p-71-iso8859-1
- b&h-lucida-medium-i-normal-sans-14-140-75-75-p-82-iso8859-1
- b&h-lucida-medium-i-normal-sans-18-180-75-75-p-105-iso8859-1
- b&h-lucida-medium-i-normal-sans-19-190-75-75-p-108-iso8859-1
- b&h-lucida-medium-i-normal-sans-24-240-75-75-p-
- b&h-lucida-medium-r-normal-sans-8-80-75-75-p-45-iso8859-1
- b&h-lucida-medium-r-normal-sans-10-100-75-75-p-58-iso8859-1
- b&h-lucida-medium-r-normal-sans-12-120-75-75-p-71-iso8859-1
- b&h-lucida-medium-r-normal-sans-14-140-75-75-p-81-iso8859-1
- b&h-lucida-medium-r-normal-sans-18-180-75-75-p-106-iso8859-1
- b&h-lucida-medium-r-normal-sans-19-190-75-75-p-108-iso8859-1
- b&h-lucida-medium-r-normal-sans-24-240-75-75-p-
- b&h-lucida-bold-i-normal-sans-8-80-75-75-p-49-iso8859-1
- b&h-lucida-bold-i-normal-sans-10-100-75-75-p-67-iso8859-1
- b&h-lucida-bold-i-normal-sans-12-120-75-75-p-79-iso8859-1
- b&h-lucida-bold-i-normal-sans-14-140-75-75-p-92-iso8859-1
- b&h-lucida-bold-i-normal-sans-18-180-75-75-p-119-iso8859-1
- b&h-lucida-bold-i-normal-sans-19-190-75-75-p-122-iso8859-1
- b&h-lucida-bold-i-normal-sans-24-240-75-75-p-
- b&h-lucida-bold-r-normal-sans-8-80-75-75-p-50-iso8859-1
- b&h-lucida-bold-r-normal-sans-10-100-75-75-p-66-iso8859-1
- b&h-lucida-bold-r-normal-sans-12-120-75-75-p-79-iso8859-1
- b&h-lucida-bold-r-normal-sans-14-140-75-75-p-92-iso8859-1
- b&h-lucida-bold-r-normal-sans-18-180-75-75-p-120-iso8859-1
- b&h-lucida-bold-r-normal-sans-19-190-75-75-p-122-iso8859-1
- b&h-lucida-bold-r-normal-sans-24-240-75-75-p-



Foundry: b&h
 Family: lucidabright

-b&h-lucidabright-medium-i-normal-8-80-75-75-p-45-iso8859-1
 -b&h-lucidabright-medium-i-normal-10-100-75-75-p-57-iso8859-1
 -b&h-lucidabright-medium-i-normal-12-120-75-75-p-67-iso8859-1
 -b&h-lucidabright-medium-i-normal-14-140-75-75-p-80-iso8859-1
 -b&h-lucidabright-medium-i-normal-18-180-75-75-p-102-iso8859-1
 -b&h-lucidabright-medium-i-normal-19-190-75-75-p-109-iso8859-1
 -b&h-lucidabright-medium-i-normal-24-240-75-75-p-114-iso8859-1
 -b&h-lucidabright-medium-r-normal-8-80-75-75-p-45-iso8859-1
 -b&h-lucidabright-medium-r-normal-10-100-75-75-p-56-iso8859-1
 -b&h-lucidabright-medium-r-normal-12-120-75-75-p-68-iso8859-1
 -b&h-lucidabright-medium-r-normal-14-140-75-75-p-80-iso8859-1
 -b&h-lucidabright-medium-r-normal-18-180-75-75-p-103-iso8859-1
 -b&h-lucidabright-medium-r-normal-19-190-75-75-p-109-iso8859-1
 -b&h-lucidabright-medium-r-normal-24-240-75-75-p-114-iso8859-1
 -b&h-lucidabright-demibold-i-normal-8-80-75-75-p-48-iso8859-1
 -b&h-lucidabright-demibold-i-normal-10-100-75-75-p-59-iso8859-1
 -b&h-lucidabright-demibold-i-normal-12-120-75-75-p-72-iso8859-1
 -b&h-lucidabright-demibold-i-normal-14-140-75-75-p-84-iso8859-1
 -b&h-lucidabright-demibold-i-normal-18-180-75-75-p-107-iso8859-1
 -b&h-lucidabright-demibold-i-normal-19-190-75-75-p-114-iso8859-1
 -b&h-lucidabright-demibold-i-normal-24-240-75-75-p-114-iso8859-1
 -b&h-lucidabright-demibold-r-normal-8-80-75-75-p-47-iso8859-1
 -b&h-lucidabright-demibold-r-normal-10-100-75-75-p-59-iso8859-1
 -b&h-lucidabright-demibold-r-normal-12-120-75-75-p-71-iso8859-1
 -b&h-lucidabright-demibold-r-normal-14-140-75-75-p-84-iso8859-1
 -b&h-lucidabright-demibold-r-normal-18-180-75-75-p-107-iso8859-1
 -b&h-lucidabright-demibold-r-normal-19-190-75-75-p-114-iso8859-1
 -b&h-lucidabright-demibold-r-normal-24-240-75-75-p-114-iso8859-1

Foundry: adobe
Family: times

- adobe-times-medium-i-normal--8-90-75-75-p-42-iso8859-1
- adobe-times-medium-i-normal--10-100-75-75-p-52-iso8859-1
- adobe-times-medium-i-normal--12-120-75-75-p-63-iso8859-1
- adobe-times-medium-i-normal--14-140-75-75-p-73-iso8859-1
- adobe-times-medium-i-normal--18-180-75-75-p-94-iso8859-1
- adobe-times-medium-i-normal--24-240-75-75-p-125-iso8859-1
- adobe-times-medium-r-normal--8-90-75-75-p-44-iso8859-1
- adobe-times-medium-r-normal--10-100-75-75-p-54-iso8859-1
- adobe-times-medium-r-normal--12-120-75-75-p-64-iso8859-1
- adobe-times-medium-r-normal--14-140-75-75-p-74-iso8859-1
- adobe-times-medium-r-normal--18-180-75-75-p-94-iso8859-1
- adobe-times-medium-r-normal--24-240-75-75-p-124-iso8859-1
- adobe-times-bold-i-normal--8-90-75-75-p-47-iso8859-1
- adobe-times-bold-i-normal--10-100-75-75-p-57-iso8859-1
- adobe-times-bold-i-normal--12-120-75-75-p-68-iso8859-1
- adobe-times-bold-i-normal--14-140-75-75-p-77-iso8859-1
- adobe-times-bold-i-normal--18-180-75-75-p-98-iso8859-1
- adobe-times-bold-i-normal--24-240-75-75-p-128-iso8859-1
- adobe-times-bold-r-normal--8-90-75-75-p-47-iso8859-1
- adobe-times-bold-r-normal--10-100-75-75-p-57-iso8859-1
- adobe-times-bold-r-normal--12-120-75-75-p-67-iso8859-1
- adobe-times-bold-r-normal--14-140-75-75-p-77-iso8859-1
- adobe-times-bold-r-normal--18-180-75-75-p-99-iso8859-1
- adobe-times-bold-r-normal--24-240-75-75-p-132-iso8859-1

Foundry: adobe
Family: symbol

-αδοβε-σψμβολ-μεδιυμ-ρ-νορμαλ--8-80-75-75-π-51-αδοβε-φοντσπεχιφιχ
-αδοβε-σψμβολ-μεδιυμ-ρ-νορμαλ--10-100-75-75-π-61-αδοβε-φοντσπεχιφιχ
-αδοβε-σψμβολ-μεδιυμ-ρ-νορμαλ--12-120-75-75-π-74-αδοβε-φοντσπεχιφιχ
-αδοβε-σψμβολ-μεδιυμ-ρ-νορμαλ--14-140-75-75-π-85-αδοβε-φοντσπεχιφιχ
-αδοβε-σψμβολ-μεδιυμ-ρ-νορμαλ--18-180-75-75-π-107-αδοβε-φοντσπεχιφιχ
-αδοβε-σψμβολ-μεδιυμ-ρ-νορμαλ--24-240-75-75-π-142-αδο

Foundry: b&h
Family: lucidatypewriter

-b&h-lucidatypewriter-medium-r-normal-sans-8-80-75-75-m-50-iso8859-1
-b&h-lucidatypewriter-medium-r-normal-sans-10-100-75-75-m-60-iso8859-1
-b&h-lucidatypewriter-medium-r-normal-sans-12-120-75-75-m-70-iso8859-1
-b&h-lucidatypewriter-medium-r-normal-sans-14-140-75-75-m-90-iso8859-1
-b&h-lucidatypewriter-medium-r-normal-sans-18-180-75-75-m-110-iso8859-1
-b&h-lucidatypewriter-medium-r-normal-sans-19-190-75-75-m-110-iso8859-1
-b&h-lucidatypewriter-medium-r-normal-sans-24-240-75-75-m-142-iso8859-1
-b&h-lucidatypewriter-bold-r-normal-sans-8-80-75-75-m-50-iso8859-1
-b&h-lucidatypewriter-bold-r-normal-sans-10-100-75-75-m-60-iso8859-1
-b&h-lucidatypewriter-bold-r-normal-sans-12-120-75-75-m-70-iso8859-1
-b&h-lucidatypewriter-bold-r-normal-sans-14-140-75-75-m-90-iso8859-1
-b&h-lucidatypewriter-bold-r-normal-sans-18-180-75-75-m-110-iso8859-1
-b&h-lucidatypewriter-bold-r-normal-sans-19-190-75-75-m-110-iso8859-1
-b&h-lucidatypewriter-bold-r-normal-sans-24-240-75-75-m-142-iso8859-1

Foundry: bitstream
Family: charter

- bitstream-charter-medium-i-normal--8-80-75-75-p-44-iso8859-1
- bitstream-charter-medium-i-normal--10-100-75-75-p-55-iso8859-1
- bitstream-charter-medium-i-normal--12-120-75-75-p-65-iso8859-1
- bitstream-charter-medium-i-normal--15-140-75-75-p-82-iso8859-1
- bitstream-charter-medium-i-normal--19-180-75-75-p-103-iso8859-1
- bitstream-charter-medium-i-normal--25-240-75-75-p-136-iso8859-1
- bitstream-charter-medium-r-normal--8-80-75-75-p-45-iso8859-1
- bitstream-charter-medium-r-normal--10-100-75-75-p-56-iso8859-1
- bitstream-charter-medium-r-normal--12-120-75-75-p-67-iso8859-1
- bitstream-charter-medium-r-normal--15-140-75-75-p-84-iso8859-1
- bitstream-charter-medium-r-normal--19-180-75-75-p-106-iso8859-1
- bitstream-charter-medium-r-normal--25-240-75-75-p-139-iso8859-1
- bitstream-charter-bold-i-normal--8-80-75-75-p-50-iso8859-1
- bitstream-charter-bold-i-normal--10-100-75-75-p-62-iso8859-1
- bitstream-charter-bold-i-normal--12-120-75-75-p-74-iso8859-1
- bitstream-charter-bold-i-normal--15-140-75-75-p-93-iso8859-1
- bitstream-charter-bold-i-normal--19-180-75-75-p-117-iso8859-1
- bitstream-charter-bold-i-normal--25-240-75-75-p-154-iso8859-1
- bitstream-charter-bold-r-normal--8-80-75-75-p-51-iso8859-1
- bitstream-charter-bold-r-normal--10-100-75-75-p-63-iso8859-1
- bitstream-charter-bold-r-normal--12-120-75-75-p-75-iso8859-1
- bitstream-charter-bold-r-normal--15-140-75-75-p-94-iso8859-1
- bitstream-charter-bold-r-normal--19-180-75-75-p-119-iso8859-1
- bitstream-charter-bold-r-normal--25-240-75-75-p-157-iso8859-1

```
-bitstream-terminal-medium-r-normal--18-140-100  
-bitstream-terminal-bold-r-normal--18-140-100-1
```

Foundry: *bitstream*
Family: *terminal*

```
.bitstream-charter-medium-r-normal--0-0-0-0-p-0-iso8859-1  
-bitstream-charter-medium-i-normal--0-0-0-0-p-0-iso8859-1  
.bitstream-charter-bold-r-normal--0-0-0-0-p-0-iso8859-1  
-bitstream-charter-bold-i-normal--0-0-0-0-p-0-iso8859-1  
-bitstream-courier-medium-r-normal--0-0-0-0-m-0-i  
-bitstream-courier-medium-i-normal--0-0-0-0-m-0-i  
-bitstream-courier-bold-r-normal--0-0-0-0-m-0-iso  
-bitstream-courier-bold-i-normal--0-0-0-0-m-0-iso
```

Foundry: *bitstream*
Families: *charter, courier*
(Scalable fonts)

```
-dec-terminal-medium-r-normal--14-140-75-75-c-80-iso8859-1  
-dec-terminal-bold-r-normal--14-140-75-75-c-80-iso8859-1
```

Foundry: *dec*
Family: *terminal*

Foundry: misc
Family: fixed

-misc-fixed-medium-r-normal--8-80-75-75-c-50-iso8859-1
-misc-fixed-medium-r-normal--9-90-75-75-c-60-iso8859-1
-misc-fixed-medium-r-normal--10-100-75-75-c-60-iso8859-1
-misc-fixed-medium-r-semicondensed--12-110-75-75-c-60-iso8859-1
-misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso8859-1
-misc-fixed-medium-r-normal--13-120-75-75-c-70-iso8859-1
-misc-fixed-medium-r-normal--13-120-75-75-c-80-iso8859-1
-misc-fixed-medium-r-normal--14-130-75-75-c-70-iso8859-1
-misc-fixed-medium-r-normal--15-140-75-75-c-90-iso8859-1
-misc-fixed-medium-r-normal--20-200-75-75-c-100-iso8859-1
~~-misc-fixed-bold-r-semicondensed--13-120-75-75-c-60-iso8859-1~~
~~-misc-fixed-bold-r-normal--13-120-75-75-c-70-iso8859-1~~
~~-misc-fixed-bold-r-normal--13-120-75-75-c-80-iso8859-1~~
~~-misc-fixed-bold-r-normal--15-140-75-75-c-90-iso8859-1~~

Foundry: schumacher
 Family: clean

```

-schumacher-clean-medium-i-normal--8-80-75-75-c-80-iso8859-1
-schumacher-clean-medium-i-normal--12-120-75-75-c-60-iso8859-1
-schumacher-clean-medium-r-normal--6-60-75-75-c-60-iso8859-1
-schumacher-clean-medium-r-normal--6-60-75-75-c-40-iso8859-1
-schumacher-clean-medium-r-normal--8-80-75-75-c-50-iso8859-1
-schumacher-clean-medium-r-normal--8-80-75-75-c-50-iso8859-1
-schumacher-clean-medium-r-normal--8-80-75-75-c-60-iso8859-1
-schumacher-clean-medium-r-normal--8-80-75-75-c-70-iso8859-1
-schumacher-clean-medium-r-normal--8-80-75-75-c-80-iso8859-1
-schumacher-clean-medium-r-normal--10-100-75-75-c-70-iso8859-1
-schumacher-clean-medium-r-normal--10-100-75-75-c-80-iso8859-1
-schumacher-clean-medium-r-normal--10-100-75-75-c-50-iso8859-1
-schumacher-clean-medium-r-normal--10-100-75-75-c-60-iso8859-1
-schumacher-clean-medium-r-normal--12-120-75-75-c-60-iso8859-1
-schumacher-clean-medium-r-normal--12-120-75-75-c-70-iso8859-1
-schumacher-clean-medium-r-normal--12-120-75-75-c-80-iso8859-1
-schumacher-clean-medium-r-normal--13-130-75-75-c-80-iso8859-1
-schumacher-clean-medium-r-normal--13-130-75-75-c-60-iso8859-1
-schumacher-clean-medium-r-normal--14-140-75-75-c-80-iso8859-1
-schumacher-clean-medium-r-normal--14-140-75-75-c-70-iso8859-1
-schumacher-clean-medium-r-normal--15-150-75-75-c-90-iso8859-1
-schumacher-clean-medium-r-normal--16-160-75-75-c-80-iso8859-1
-schumacher-clean-bold-r-normal--8-80-75-75-c-80-iso8859-1
-schumacher-clean-bold-r-normal--10-100-75-75-c-60-iso8859-1
-schumacher-clean-bold-r-normal--10-100-75-75-c-80-iso8859-1
-schumacher-clean-bold-r-normal--12-120-75-75-c-80-iso8859-1
-schumacher-clean-bold-r-normal--12-120-75-75-c-60-iso8859-1
-schumacher-clean-bold-r-normal--13-130-75-75-c-80-iso8859-1
-schumacher-clean-bold-r-normal--14-140-75-75-c-80-iso8859-1
-schumacher-clean-bold-r-normal--15-150-75-75-c-90-iso8859-1
-schumacher-clean-bold-r-normal--16-160-75-75-c-80-iso8859-1

```


	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
'	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{		}	~	
	ı	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
ö	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Encoding: iso8859

	◆	⌘	H	F	C	L	°	±	N	V	J	γ	Γ	L	†
-	-	-	-	-	†	†	⊥	τ		≦	≧	π	≠	£	·
	!	”	#	\$	%	&	'	()	*	+	,	-	·	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[¥]	^	_
'	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{		}	~	
	。	「	」	、	・	ヲ	ア	イ	ウ	エ	オ	カ	キ	ク	ケ
-	ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ	サ	シ	ス	セ	ソ
夕	チ	ツ	テ	ト	ナ	ニ	ヌ	ネ	ノ	ハ	ヒ	フ	ヘ	ホ	マ
ミ	ム	メ	モ	ヤ	ユ	ヨ	ラ	リ	ル	レ	ロ	ワ	ン	”	°

Encoding: jisx0201.1976

	◆	■									「	」	レ	ル	ト
-	-	-	-	-	ト	↓	⊥	⊥							
	!	”	#	\$	%	&	'	()	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[¥]	^	_
'	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{		}	-	
	。	「	」	、	・	ヲ	フ	イ	ウ	エ	オ	ヤ	ユ	ヨ	ツ
-	ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ	サ	シ	ス	セ	ソ
々	チ	ツ	テ	ト	ナ	ニ	ヌ	ネ	ノ	ハ	ヒ	フ	ヘ	ホ	マ
々	ム	メ	モ	ヤ	ユ	ヨ	ラ	リ	ル	レ	ロ	ワ	ヅ	”	°

Encoding: jisx0201.1976

Encoding: SunOLcursor

Example B-1 is the source code for the *xshowfonts* program, which we used to create most of the illustrations in this appendix. If you don't want to type it in, you can find instructions for getting it online in Appendix H, *Obtaining Example Programs*.

Example B-1. xshowfont source listing

```
/* Dan Heller <argv@sun.com>, based on a design by Tim O'Reilly
 *
 * xshowfonts.c -
 * Displays a set of fonts specified on the command line, from
 * a pipe, or typed into stdin. Fonts can be specified as specific
 * or wildcard character strings. A pixmap is created to
 * display all the fonts. This is done by using the pixmap as the
 * pixmap image for a label widget. Each font prints its own name
 * in its own font style -- the -phrase option prints the phrase
 * instead.
 *
 * All fonts are loaded first and scanned to determine the total
 * width and height of the pixmap first. Then the fonts are
 * reopened again to actually render the fonts into the pixmap.
 * All this could be avoided by using XListFontsWithInfo()
 * rather than XListFonts() but since the list is potentially
 * very large, I didn't want to overload the server and client
 * with all those fonts + a very large pixmap.
 *
 * Usage: xshowfonts
 * -s sorts the fonts in alphabetical order before displaying
 * them.
 * -v verbose mode for when input is redirected to stdin.
 * -w width of viewport window
 * -h height of viewport window
 * -fg foreground_color
 * -bg background_color
 * -phrase "text string" (otherwise, name of font is used)
 * -indicates to read from stdin. Piping doesn't require
 * the '-' argument. With no arguments, xshowfonts reads
 * from stdin anyway.
 *
 * Neat ways to use the program:
 * xshowfonts -fg green -bg black "**adobe*"
 * xshowfonts -sort "*"
 * xshowfonts -phrase "The quick brown fox jumps over the lazy
 * dog" "**times*"
 * xlsfonts | xshowfonts -sort
 * xshowfonts "**helvetica*"
 *
 * compile: (triple click and paste next line)
 * cc -O -s xshowfonts.c -lXaw -lXt -lXmu -lX11 -o xshowfonts
 */

#include <stdio.h>
#include <X11/Intrinsic.h>
#include <X11/StringDefs.h>
#include <X11/Xaw/Label.h>
#include <X11/Xaw/Viewport.h>

struct _resrcs {
    int sort;
```

Example B-1. xshowfont source listing (continued)

```
int verbose;
Pixel fg, bg;
char *phrase;
int view_width, view_height;
} Resrcs;

static XtResource resources[] = {
    { "sort", "Sort", XtRBoolean, sizeof (int),
      XtOffsetOf(struct _resrcs,sort), XtRImmediate,
      False },
    { "verbose", "Verbose", XtRBoolean, sizeof (int),
      XtOffsetOf(struct _resrcs,verbose), XtRImmediate,
      False },
    { "foreground", "Foreground", XtRPixel, sizeof (Pixel),
      XtOffsetOf(struct _resrcs,fg), XtRString,
      XtDefaultForeground },
    { "background", "Background", XtRPixel, sizeof (Pixel),
      XtOffsetOf(struct _resrcs,bg), XtRString,
      XtDefaultBackground },
    { "phrase", "Phrase", XtRString, sizeof (String),
      XtOffsetOf(struct _resrcs,phrase), XtRImmediate, NULL },
    { "view-width", "View-width", XtRInt, sizeof (int),
      XtOffsetOf(struct _resrcs,view_width), XtRImmediate,
      (char *)500 },
    { "view-height", "View-height", XtRInt, sizeof (int),
      XtOffsetOf(struct _resrcs,view_height), XtRImmediate,
      (char *)300 },
};

static XrmOptionDescRec options[] = {
    { "-sort", "sort", XrmoptionNoArg, "True" },
    { "-v", "verbose", XrmoptionNoArg, "True" },
    { "-fg", "foreground", XrmoptionSepArg, NULL },
    { "-bg", "background", XrmoptionSepArg, NULL },
    { "-phrase", "phrase", XrmoptionSepArg, NULL },
    { "-w", "view-width", XrmoptionSepArg, NULL },
    { "-h", "view-height", XrmoptionSepArg, NULL },
};

/* sort font according to these parameters.
 * font specs we're interested in:
 * -fndry-fmly-wght-slant-*swdth-*adstyl-*pxlsz-ptsz- ....
 * foundry -- sort by foundry first; similar ones are always
 * grouped together
 * weight -- medium, demi-bold, bold
 * slant -- roman, italic/oblique, reverse italic/oblique
 * (i or o, r, ri, ro)
 * ptsize -- increase numerical order
 */
font_cmp(f1, f2)
char **f1, **f2;
{
    char fndry1[16], fmly1[64], wght1[32], slant1[3];
    char fndry2[16], fmly2[64], wght2[32], slant2[3];
    int n, m, ptsize1, ptsize2;
    char *font_fmt_str = "-%[^-]-%[^-]-%[^-]-%[^-]-%*[^0-9]%
        *d-%d-";
}
```

Example B-1. xshowfont source listing (continued)

```
n = sscanf(*f1, font_fmt_str, fndry1, fmly1, wght1, slant1,
          &ptsize1);
m = sscanf(*f2, font_fmt_str, fndry2, fmly2, wght2, slant2,
          &ptsize2);
if (m < 5 || n < 5)
    /* font not in correct format -- just return font names
     * in order */
    return strcmp(*f1, *f2);
if (n = strcmp(fndry1, fndry2))
    return n; /* different foundries -- return alphabetical
             * order */
if (n = strcmp(fmly1, fmly2))
    return n; /* different families -- return alphabetical
             * order */
if (n = strcmp(wght1, wght2))
    return -n; /* weight happens to be correct in reverse
              * alpha order */
if (n = strcmp(slant1, slant2))
    return n; /* slants happen to be correct in alphabetical
             * order */
/* sort according to point size */
return psize1 - psize2;
}

main(argc, argv)
int argc;
char *argv[];
{
    Widget topLevel, vp;
    char **list = (char **)NULL, **tmp;
    char buf[128];
    extern char **XListFonts();
    extern int strcmp();
    XFontStruct *font;
    Pixmap pixmap;
    GC gc;
    Display *dpy;
    int istty = isatty(0), redirect = !istty, i, j, total = 0;
    unsigned int w, width = 0, height = 0;

    topLevel = XtInitialize(argv[0], argv[0], options,
        XtNumber(options), &argc, argv);
    dpy = XtDisplay(topLevel);

    XtGetApplicationResources(topLevel, &Resrcs,
        resources, XtNumber(resources), NULL, 0);

    if (!argv[1] || !strcmp(argv[1], "-")) {
        printf("Loading fonts from input. ");
        if (istty) {
            puts("End with EOF or .");
            redirect++;
        } else
            puts("Use -v to view font names being loaded.");
    } else if (!istty && strcmp(argv[1], "-"))
        printf("%s: either use pipes or specify font names --
            not both.\n",
            argv[0]), exit(1);
}
```

Example B-1. xshowfont source listing (continued)

```
while (*++argv || redirect) {
    if (!redirect)
        if (!strcmp(*argv, "-"))
            redirect++;
        else
            strcpy(buf, *argv);
    if (redirect) {
        if (istty)
            printf("Fontname: "), fflush(stdout);
        if (!fgets(buf, sizeof buf, stdin) ||
            !strcmp(buf, ".\n"))
            break;
        buf[strlen(buf)-1] = 0;
    }
    if (!buf[0])
        continue;
    if (istty || Resrcs.verbose)
        printf("Loading
tmp = XListFonts(dpy, buf, 32767, &i);
if (i == 0) {
    printf("couldn't load font ");
    if (!istty && !Resrcs.verbose)
        printf("
    putchar('\n');
    continue;
}
if (istty || Resrcs.verbose)
    printf("%d font%s\n", i, i == 1? "" : "s");
if (!list) {
    list = tmp;
    total = i;
} else {
    i += total;
    if (!(list = (char **)XtRealloc(list, i *
        sizeof (char *))))
        XtError("Not enough memory for font names");
    for (j = 0; total < i; j++, total++)
        list[total] = tmp[j];
}
}
if (total == 0)
    puts("No fonts?!"), exit(1);
printf("Total fonts loaded: %d\n", total);
if (Resrcs.sort) {
    printf("Sorting fonts..."), fflush(stdout);
    qsort(list, total, sizeof (char *), font_cmp);
    putchar('\n');
}
/* calculate size for pixmap by getting the dimensions
 * of each font */
puts("Calculating sizes for pixmap.");
for (i = 0; i < total; i++) {
    if (!(font = XLoadQueryFont(dpy, list[i]))) {
        printf("Can't load font: %s\n", list[i]);
        continue;
    }
}
```

Example B-1. xshowfont source listing (continued)

```
    if ((w = XTextWidth(font, list[i],
        strlen(list[i]))) > width)
        width = w;
    height += font->ascent + font->descent;
    XFreeFont(dpy, font);
}
width += 6;
height += 6;
/* Create pixmap + GC */
printf("Creating pixmap of size %dx%d\n", width, height);
if (!(pixmap = XCreatePixmap(dpy, DefaultRootWindow(dpy),
    width, height, DefaultDepth(dpy, DefaultScreen(dpy)))))
    XtError("Can't Create pixmap");
if (!(gc = XCreateGC(dpy, pixmap, NULL, 0)))
    XtError("Can't create gc");
XSetForeground(dpy, gc, Resrcs.bg);
XFillRectangle(dpy, pixmap, gc, 0, 0, width, height);
XSetForeground(dpy, gc, Resrcs.fg);
XSetBackground(dpy, gc, Resrcs.bg);
height = 0;
for (i = 0; i < total; i++) {
    if (!(font = XLoadQueryFont(dpy, list[i])))
        continue; /* it's already been reported */
    XSetFont(dpy, gc, font->fid);
    height += font->ascent;
    if (Resrcs.phrase)
        XDrawString(dpy, pixmap, gc, 0, height,
            Resrcs.phrase, strlen(Resrcs.phrase));
    else
        XDrawString(dpy, pixmap, gc, 5, height, list[i],
            strlen(list[i]));
    height += font->descent;
    XFreeFont(dpy, font);
}
vp = XtVaCreateManagedWidget("viewport", viewportWidgetClass,
    topLevel,
    XtNallowHoriz, True,
    XtNallowVert, True,
    XtNwidth, Resrcs.view_width,
    XtNheight, Resrcs.view_height,
    NULL);
XtVaCreateManagedWidget("_foo", labelWidgetClass, vp,
    XtNbitmap, pixmap,
    NULL);

if (!redirect)
    XFreeFontNames(list);

XtRealizeWidget(topLevel);
XtMainLoop();
}
```


C

Standard Bitmaps

This appendix shows the bitmaps included with the standard distribution of the X Window System. These can be used for setting window backgrounds, cursor symbols, pixmaps, and possibly for application icon pixmaps.

C

Standard Bitmaps

A number of bitmaps are included with the standard distribution of the X Window System. These bitmaps can be used for setting window backgrounds, cursor symbols, pixmaps, and possibly for application icon pixmaps.

The standard bitmaps are generally located in the directory `/usr/include/X11/bitmaps`. Each bitmap is in standard X11 bitmap format in its own file. The `bitmap` application can be used to view these bitmaps in larger scale and to edit them (though their permissions normally do not allow overwriting).

You can use these bitmaps to set the background pattern of a window in any application that allows it. For example, if you wanted to change the root window background, you could do so using `xsetroot`:

```
xsetroot -bitmap /usr/include/X11/bitmaps/wide_weave
```

See Chapter 14, *Setup Clients*, for more information about `xsetroot`.

Note that the bitmaps that come in pairs, such as `cntr_ptr` and `cntr_ptrmsk`, are intended for creating pointer shapes. For information on specifying a bitmap as the root window pointer using `xsetroot`, see Chapter 13.

The 86 bitmaps pictured on the following pages are included in the Release 5 standard distribution of X. The following 23 bitmaps have been added to the standard distribution in Release 5.

Table C-1. Standard Bitmaps Added in Release 5

Dashes	RotateLeft	grid2
Down	RotateRight	grid4
Excl	Stipple	grid8
FlipHoriz	Term	ldblarrow
FlipVert	Up	menu6
Fold	black6	rdblarrow
Left	box6	stripe4
Right	grid16	














































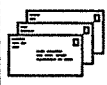


































1x1 	2x2 	Dashes 	Down 	Excl 
FlipHoriz 	FlipVert 	Fold 	Left 	Right 
RotateLeft 	RotateRight 	Stipple 	Term 	Up 
black 	black6 	box6 	boxes 	calculator 
cntr_ptr 	cntr_ptrmsk 	cross_weave 	dimple1 	dimple3 
dot 	dropbar7 	dropbar8 	flagdown 	flagup 
flipped_gray 	gray 	gray1 	gray3 	grid16 
grid2 	grid4 	grid8 	hlines2 	hlines3 
icon 	keyboard16 	ldblarrow 	left_ptr 	left_ptrmsk 
letters 	light_gray 	mailempty 	mailemptymask 	mailfull 

Figure C-1. The standard bitmaps

mailfullmsk 	menu10 	menu12 	menu16 	menu6 
menu8 	noletters 	opendot 	opendotMask 	plaid 
rdblarrow 	right_ptr 	right_ptmsk 	root_weave 	scales 
sipb ΣΠΒ ΣΠΒ	star 	starMask 	stipple 	target 
terminal 	tie_fighter 	vlines2 	vlines3 	weird_size 
wide_weave 	wingdogs 	xfd icon A B C D E F	xlogo11 	xlogo16 
xlogo32 	xlogo64 			

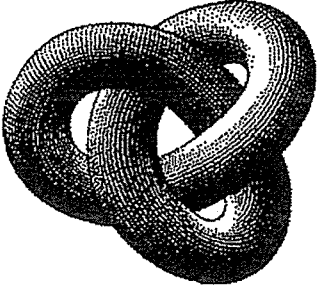
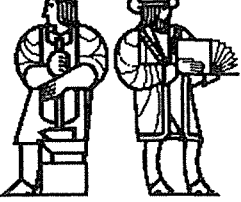

escherknot 	mensetmanus 	woman 
---	--	--

Figure C-1. The Standard Bitmaps (continued)

D

Standard Cursors

This appendix shows the standard cursor images that can be used by X programs.

D

Standard Cursors

Table D-1 lists the cursors available in the standard distribution of X from MIT; the cursor shapes themselves are pictured in Figure D-1.

To specify a cursor as an argument to a command-line option, as the value of a resource variable, etc., strip the `XC_` prefix from the symbol name. For example, to specify the `XC_sailboat` cursor as the *xterm* pointer, you could enter the command:

```
% xterm -xrm 'xterm*pointerShape: sailboat'
```

Each cursor has an associated numeric value (to the right of the symbol name in the table). You may notice that the values skip the odd numbers. Each cursor is actually composed of two font characters: the character that defines the shape (pictured in Figure D-0), and a mask character (not shown) that sets the cursor shape off from the root (or other) window. (More precisely, the mask selects which pixels in the screen around the cursor are disturbed by the cursor.) The mask is generally the same shape as the character it underlies but is one pixel wider in all directions. The even numbers in the table actually correspond to the cursor's mask character.

To get an idea of what masks look like, display the entire cursor font using the command:

```
% xfd -fn cursor
```

The *mwm* window manager uses several of the standard cursor symbols. In addition, *mwm* uses some Motif-specific cursors, which are illustrated in Figure 1-3 in Part One of this guide.

Table D-1. Standard Cursor Symbols

Symbol	Value	Symbol	Value
XC_X_cursor	0	XC_ll_angle	76
XC_arrow	2	XC_lr_angle	78
XC_based_arrow_down	4	XC_man	80
XC_based_arrow_up	6	XC_middlebutton	82
XC_boat	8	XC_mouse	84
XC_bogosity	10	XC_pencil	86
XC_bottom_left_corner	12	XC_pirate	88
XC_bottom_right_corner	14	XC_plus	90
XC_bottom_side	16	XC_question_arrow	92
XC_bottom_tee	18	XC_right_ptr	94
XC_box_spiral	20	XC_right_side	96
XC_center_ptr	22	XC_right_tee	98
XC_circle	24	XC_rightbutton	100
XC_clock	26	XC_rtl_logo	102
XC_coffee_mug	28	XC_sailboat	104
XC_cross	30	XC_sb_down_arrow	106
XC_cross_reverse	32	XC_sb_h_double_arrow	108
XC_crosshair	34	XC_sb_left_arrow	110
XC_diamond_cross	36	XC_sb_right_arrow	112
XC_dot	38	XC_sb_up_arrow	114
XC_dotbox	40	XC_sb_v_double_arrow	116
XC_double_arrow	42	XC_shuttle	118
XC_draft_large	44	XC_sizing	120
XC_draft_small	46	XC_spider	122
XC_draped_box	48	XC_spraycan	124
XC_exchange	50	XC_star	126
XC_fleur	52	XC_target	128
XC_gobbler	54	XC_tcross	130
XC_gumby	56	XC_top_left_arrow	132
XC_hand1	58	XC_top_left_corner	134
XC_hand2	60	XC_top_right_corner	136
XC_heart	62	XC_top_side	138
XC_icon	64	XC_top_tee	140
XC_iron_cross	66	XC_trek	142
XC_left_ptr	68	XC_ul_angle	144
XC_left_side	70	XC_umbrella	146
XC_left_tee	72	XC_ur_angle	148
XC_leftbutton	74	XC_watch	150
		XC_xterm	152

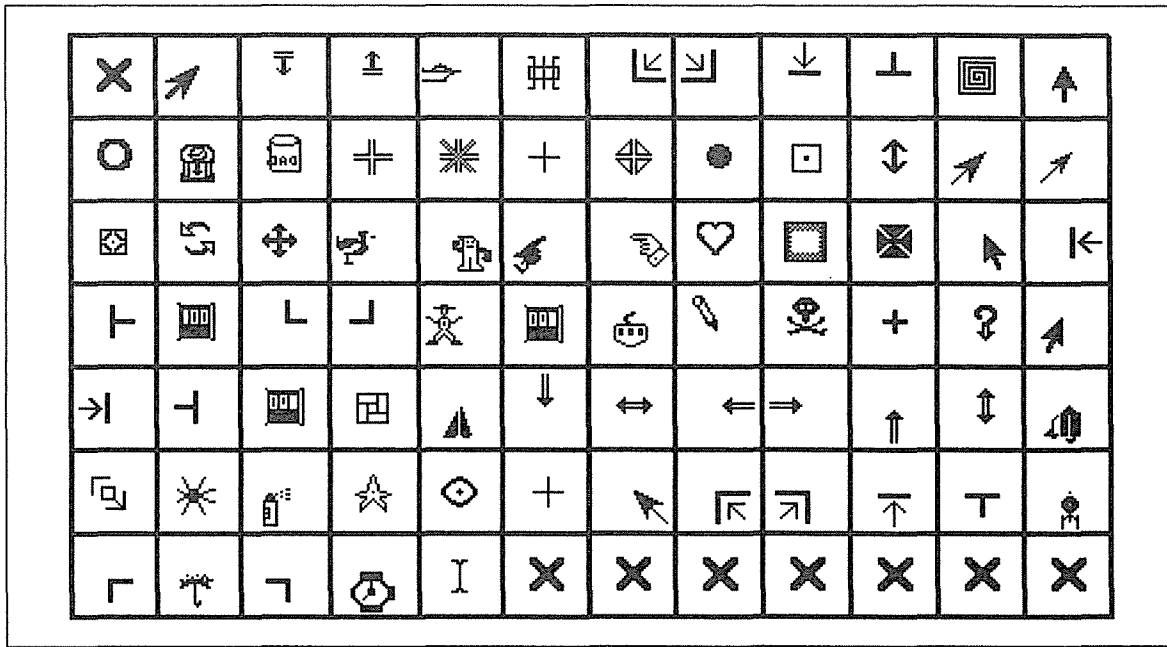


Figure D-1. The standard cursors

E

xterm Control Sequences

This appendix list the escape sequences that can be used to control features of an xterm window or its terminal emulation.

In This Appendix:

Definitions	807
VT100 Mode	808
Mouse Tracking	813
Tektronix 4014 Mode	814

E

xterm Control Sequences

A standard terminal performs many operations in response to escape sequences sent out by a program: redraws the screen, backspaces, advances a line, etc. Under UNIX, programs use the *termcap* or *terminfo* database to determine which escape sequences to send out. (For more information, see the standard UNIX man pages *termcap(5)* or *terminfo(5)*, or the Nutshell Handbook *Termcap and Terminfo*, available from O'Reilly & Associates, Inc.)

In emulating a terminal, *xterm* responds to those same terminal escape sequences. This appendix lists the valid escape sequences for *xterm*. Although these sequences are primarily intended to be used by a program running in the *xterm* window, be aware that a user can affect the window's operation by sending it an escape sequence using the UNIX *echo(1)* command. Chapter 6, *Font Specification*, describes the use of an escape sequence to change the display font dynamically—a typical and useful example.

This appendix is based on two sources: the “Xterm Control Sequences” document, written by Edward Moy, University of California, Berkeley, for the X10 *xterm*; and revisions for X11 R5 provided by Stephen Gildea of the MIT X Consortium.

Definitions

- Ⓢ The literal character c.
- C A single (required) character.
- P_s A single (usually optional) numeric parameter, composed of one or more digits.
- P_m A multiple numeric parameter composed of any number of single numeric parameters, separated by Ⓢ character(s).
- P_t A text parameter composed of printable characters.

VT100 Mode

Most of these control sequences are standard VT102 control sequences, but there are some sequences here from later DEC VT terminals too. Major VT102 features not supported are smooth scrolling, double size characters.

There are additional functions to provide control of *xterm*-dependent functions, such as the scrollbar or window size.

<code>BEL</code>	Bell (Ctrl-G)
<code>BS</code>	Backspace (Ctrl-H)
<code>TAB</code>	Horizontal Tab (HT) (Ctrl-I)
<code>LF</code>	Line Feed or New Line (NL) (Ctrl-J)
<code>VT</code>	Vertical Tab (Ctrl-K) (Same as LF)
<code>FF</code>	Form Feed or New Page (NP) (Ctrl-L) (Same as LF)
<code>CR</code>	Carriage Return (Ctrl-M)
<code>SO</code>	Shift Out (Ctrl-N) → Switch to Alternate Character Set (Invokes the G1 character set)
<code>SI</code>	Shift In (Ctrl-O) → Switch to Standard Character Set (Invokes the G0 character set—the default)
<code>ESC [(C</code>	Select G0 Character Set (ISO 2022) C = <code>[0</code> → DEC Special Character and Line Drawing Set C = <code>[A</code> → United Kingdom (UK) C = <code>[B</code> → United States (USASCII)
<code>ESC [) C</code>	Select G1 Character Set (ISO 2022) C = <code>[0</code> → DEC Special Character and Line Drawing Set C = <code>[A</code> → United Kingdom (UK) C = <code>[B</code> → United States (USASCII)
<code>ESC [* C</code>	Select G2 Character Set (ISO 2022) C = <code>[0</code> → DEC Special Character and Line Drawing Set C = <code>[A</code> → United Kingdom (UK) C = <code>[B</code> → United States (USASCII)
<code>ESC [+ C</code>	Select G3 Character Set (ISO 2022) C = <code>[0</code> → DEC Special Character and Line Drawing Set C = <code>[A</code> → United Kingdom (UK) C = <code>[B</code> → United States (USASCII)
<code>ESC [7</code>	Save Cursor (DECSC)
<code>ESC [8</code>	Restore Cursor (DECRC)
<code>ESC [=</code>	Application Keypad (DECPAM)
<code>ESC [></code>	Normal Keypad (DECPNM)

<code>ESC D</code>	Index (IND)
<code>ESC E</code>	Next Line (NEL)
<code>ESC H</code>	Tab Set (HTS)
<code>ESC M</code>	Reverse Index (RI)
<code>ESC N</code>	Single Shift Select of G2 Character Set (SS2) (affects next character only)
<code>ESC O</code>	Single Shift Select of G3 Character Set (SS3) (affects next character only)
<code>ESC P_t ESC</code>	Device Control String (DCS); <i>xterm</i> implements no DCS functions; P_t is ignored; P_t need not be printable characters.
<code>ESC</code>	Return Terminal ID (DECID); obsolete form of <code>ESC [I c</code> (DA)
<code>ESC [P_s @</code>	Insert P_s (Blank) Character(s) (default = 1) (ICH)
<code>ESC [P_s A</code>	Cursor Up P_s Times (default = 1) (CUU)
<code>ESC [P_s B</code>	Cursor Down P_s Times (default = 1) (CUD)
<code>ESC [P_s C</code>	Cursor Forward P_s Times (default = 1) (CUF)
<code>ESC [P_s D</code>	Cursor Backward P_s Times (default = 1) (CUB)
<code>ESC [P_s ; P_s H</code>	Cursor Position [row;column] (default = [1,1]) (CUP)
<code>ESC [P_s J</code>	Erase in Display (ED) $P_s = 0$ → Clear Below (default) $P_s = 1$ → Clear Above $P_s = 2$ → Clear All
<code>ESC [P_s K</code>	Erase in Line (EL) $P_s = 0$ → Clear to Right (default) $P_s = 1$ → Clear to Left $P_s = 2$ → Clear All
<code>ESC [P_s L</code>	Insert P_s Line(s) (default = 1) (IL)
<code>ESC [P_s M</code>	Delete P_s Line(s) (default = 1) (DL)
<code>ESC [P_s P</code>	Delete P_s Character(s) (default = 1) (DCH)
<code>ESC [P_s ; P_s ; P_s ; P_s ; T</code>	Initiate Hilite mouse tracking. Parameters are [func;startx;starty;firstrow;lastrow]. See "Mouse Tracking."
<code>ESC [P_s c</code>	Send Device Attributes (DA); $P_s = 0$ or omitted → request attributes from terminal → <code>ESC [? 1 ; 2 c</code> ("I am a VT100 with Advanced Video Option.")

<code>ESC [P_s ; P_s f</code>	Horizontal and Vertical (Cursor) Position [row;column] (default = [1,1]) (HVP)
<code>ESC [P_s g</code>	Tab Clear (TBC) $P_s = 0$ → Clear Current Column (default) $P_s = 3$ → Clear All
<code>ESC [P_m h</code>	Set Mode (SM) $P_s = 4$ → Insert Mode (IRM) $P_s = 2 0$ → Automatic Newline (LNM)
<code>ESC [P_m l</code>	Reset Mode (RM) $P_s = 4$ → Insert Mode (IRM) $P_s = 2 0$ → Automatic Linefeed (LNM)
<code>ESC [P_m m</code>	Character Attributes (SGR) $P_s = 0$ → Normal (default) $P_s = 1$ → Bold $P_s = 4$ → Underscore $P_s = 5$ → Blink (appears as Bold) $P_s = 7$ → Inverse
<code>ESC [P_s n</code>	Device Status Report (DSR) $P_s = 5$ → Status Report <code>ESC [0 n</code> → OK $P_s = 6$ → Report Cursor Position (CPR) [row;column] as <code>ESC [r ; c R</code>
<code>ESC [P_s ; P_s r</code>	Set Scrolling Region [top;bottom] (default = full size of window) (DECSTBM)
<code>ESC [P_s x</code>	Request Terminal Parameters (DECREQTPARM)
<code>ESC [? P_m h</code>	DEC Private Mode Set (DECSET) $P_s = 1$ → Application Cursor Keys (DECCKM) $P_s = 2$ → Designate USASCII for character sets G0-G3. (In the VT102, this selects VT52 mode (DECANM), which <i>xterm</i> doesn't support.) $P_s = 3$ → 132 Column Mode (DECCOLM) $P_s = 4$ → Smooth (Slow) Scroll (DECSCLM) $P_s = 5$ → Reverse Video (DECSCNM) $P_s = 6$ → Origin (Cursor) Mode (DECOM) $P_s = 7$ → Wraparound Mode (DECAWM) $P_s = 8$ → Auto-repeat Keys (DECARM) $P_s = 9$ → Send Mouse Row & Column (X & Y) on button press (see "Mouse Tracking") $P_s = 3 8$ → Enter TekTronix Mode (DECTEK) $P_s = 4 0$ → Allow 80 ↔ 132 Mode $P_s = 4 1$ → <i>curses(5)</i> fix $P_s = 4 4$ → Turn On Margin Bell $P_s = 4 5$ → Reverse-wraparound Mode $P_s = 4 6$ → Start Logging

$P_s = \boxed{4}\boxed{7} \rightarrow$ Use Alternate Screen Buffer (unless disabled by the `titleInhibit` resource)

$P_s = \boxed{1}\boxed{0}\boxed{0}\boxed{0} \rightarrow$ Send Mouse Row & Column (X & Y) on button press and release; see “Mouse Tracking”

$P_s = \boxed{1}\boxed{0}\boxed{0}\boxed{1} \rightarrow$ Use Hilite Mouse Tracking; see “Mouse Tracking”

$\boxed{\text{ESC}}\boxed{1}\boxed{?}P_m\boxed{1}$

DEC Private Mode Reset (DECRST)

$P_s = \boxed{1} \rightarrow$ Normal Cursor Keys (DECCKM)

$P_s = \boxed{3} \rightarrow$ 80 Column Mode (DECCOLM)

$P_s = \boxed{4} \rightarrow$ Jump (Fast) Scroll (DECSCLM)

$P_s = \boxed{5} \rightarrow$ Normal Video (DECSCNM)

$P_s = \boxed{6} \rightarrow$ Normal Cursor Mode (DECOM)

$P_s = \boxed{7} \rightarrow$ No Wraparound Mode (DECAWM)

$P_s = \boxed{8} \rightarrow$ No Auto-repeat Keys (DECARM)

$P_s = \boxed{9} \rightarrow$ Don't Send Mouse Row & Column (X & Y) on button press

$P_s = \boxed{4}\boxed{0} \rightarrow$ Disallow 80 \leftrightarrow 132 Mode

$P_s = \boxed{4}\boxed{1} \rightarrow$ No *curses(5)* fix

$P_s = \boxed{4}\boxed{4} \rightarrow$ Turn Off Margin Bell

$P_s = \boxed{4}\boxed{5} \rightarrow$ No Reverse-wraparound Mode

$P_s = \boxed{4}\boxed{6} \rightarrow$ Stop Logging

$P_s = \boxed{4}\boxed{7} \rightarrow$ Use Normal Screen Buffer

$P_s = \boxed{1}\boxed{0}\boxed{0}\boxed{0} \rightarrow$ Don't send Mouse Row & Column (X & Y) on button press

$P_s = \boxed{1}\boxed{0}\boxed{0}\boxed{1} \rightarrow$ Don't use/use Hilite Mouse Tracking

$\boxed{\text{ESC}}\boxed{1}\boxed{?}P_s\boxed{r}$

Restore DEC Private Mode values. The value of P_s previously saved is restored. (P_s values are the same as for DECSET. This escape sequence toggles between DECSET and DECRST.)

$P_s = \boxed{1} \rightarrow$ Normal/Application Cursor Keys (DECCKM)

$P_s = \boxed{3} \rightarrow$ 80/132 Column Mode (DECCOLM)

$P_s = \boxed{4} \rightarrow$ Jump (Fast)/Smooth (Slow) Scroll (DECSCLM)

$P_s = \boxed{5} \rightarrow$ Normal/Reverse Video (DECSCNM)

$P_s = \boxed{6} \rightarrow$ Normal/Origin Cursor Mode (DECOM)

$P_s = \boxed{7} \rightarrow$ No Wraparound/Wraparound Mode (DECAWM)

$P_s = \boxed{8} \rightarrow$ Auto-repeat/No Auto-repeat Keys (DECARM)

$P_s = \boxed{9} \rightarrow$ Don't send/send Mouse Row & Column (X & Y) on button press

$P_s = \boxed{4}\boxed{0} \rightarrow$ Disallow/Allow 80 \leftrightarrow 132 Mode

$P_s = \boxed{4}\boxed{1} \rightarrow$ Off/On *curses(5)* fix

$P_s = \boxed{4}\boxed{4} \rightarrow$ Turn Off/On Margin Bell

$P_s = \boxed{4}\boxed{5} \rightarrow$ No Reverse-wraparound/Reverse-wraparound Mode

$P_s = \boxed{4}\boxed{6} \rightarrow$ Stop/Start Logging

$P_s = \boxed{4}\boxed{7} \rightarrow$ Use Normal/Alternate Screen Buffer

$P_s = \boxed{1}\boxed{0}\boxed{0}\boxed{0} \rightarrow$ Don't send/send Row & Column (X & Y) on button press and release

$P_s = \boxed{1}\boxed{0}\boxed{0}\boxed{1} \rightarrow$ Don't use/use Hilite Mouse Tracking

`ESC [[?] Pm [S]`

Save DEC Private Mode values. P_s values are the same as for DECSET.

$P_s = [1]$ → Normal/Application Cursor Keys (DECCKM)

$P_s = [3]$ → 80/132 Column Mode (DECCOLM)

$P_s = [4]$ → Jump (Fast)/Smooth (Slow) Scroll (DECSCLM)

$P_s = [5]$ → Normal/Reverse Video (DECSCNM)

$P_s = [6]$ → Normal/Origin Cursor Mode (DECOM)

$P_s = [7]$ → No Wraparound/Wraparound Mode (DECAWM)

$P_s = [8]$ → Auto-repeat/No Auto-repeat Keys (DECARM)

$P_s = [9]$ → Don't send/send Mouse Row & Column (X & Y) on
button press

$P_s = [4][0]$ → Disallow/Allow 80 ↔ 132 Mode

$P_s = [4][1]$ → Off/On *curses(5)* fix

$P_s = [4][4]$ → Turn Off/On Margin Bell

$P_s = [4][5]$ → No Reverse-wraparound/Reverse-wraparound Mode

$P_s = [4][6]$ → Stop/Start Logging

$P_s = [4][7]$ → Use Normal/Alternate Screen Buffer

$P_s = [1][0][0][0]$ → Don't send/send Mouse Row & Column (X & Y)
on button press and release

$P_s = [1][0][0][1]$ → Don't use/use Hilite Mouse Tracking

`ESC [] Ps [;] Pt [BEL]`

Set Text Parameters

$P_s = [0]$ → Change Window/Icon Name and Window Title to P_t

$P_s = [1]$ → Change Window/Icon Name to P_t

$P_s = [2]$ → Change Window Title to P_t

$P_s = [4][6]$ → Change Log File to P_t (normally disabled by a compile-time option)

$P_s = [5][0]$ → Change Font to P_t

`ESC [Pt ESC`

Privacy Message (PM)

xterm implements no PM functions; P_t is ignored. P_t need not be printable characters.

`ESC [Pt ESC`

Application Program Command (APC)

xterm implements no APC functions; P_t is ignored. P_t need not be printable characters.

`ESC [c`

Full Reset (RIS)

`ESC [n`

Locking Shift Select of G2 Character Set (LS2)

`ESC [o`

Locking Shift Select of G3 Character Set (LS3)

`ESC [l`

Invoke the G3 Character Set as GR (LS3R). Has no visible effect in *xterm*.

`ESC [j`

Invoke the G2 Character Set as GR (LS2R). Has no visible effect in *xterm*.

`ESC [~`

Invoke the G1 Character Set as GR (LS1R). Has no visible effect in *xterm*.

Mouse Tracking

The VT widget can be set to send the mouse position and other information on button presses. These modes are typically used by editors and other full-screen applications that want to make use of the mouse.

There are three mutually exclusive modes, each enabled (or disabled) by a different parameter in the DECSET (or DECRST) escape sequence. Parameters for all mouse tracking escape sequences generated by *xterm* encode numeric parameters in a single character as *value*+040. For example, is 1. The screen coordinate system is 1-based.

X10 compatibility mode sends an escape sequence on button press encoding the location and the mouse button pressed. It is enabled by specifying parameter 9 to DECSET. On button press, *xterm* sends `ESC[M CbCxCy` (6 characters). *C_b* is button-1. *C_x* and *C_y* are the x and y coordinates of the mouse when the button was pressed.

Normal tracking mode sends an escape sequence on both button press and release. Modifier information is also sent. It is enabled by specifying parameter 1000 to DECSET. On button press or release, *xterm* sends `ESC[M CbCxCy`. The low two bits of *C_b* encode button information: 0=MB1 pressed, 1=MB2 pressed, 2=MB3 pressed, 3=release. The upper bits encode what modifiers were down when the button was pressed and are added together. 4=Shift, 8=Meta, 16=Control. *C_x* and *C_y* are the x and y coordinates of the mouse event. The upper-left corner is (1,1).

Mouse hilite tracking notifies a program of a button press, receives a range of lines from the program, highlights the region covered by the mouse within that range until button release, and then sends the program the release coordinates. It is enabled by specifying parameter 1001 to DECSET. Warning: use of this mode requires a cooperating program or it will hang *xterm*. On button press, the same information as for normal tracking is generated; *xterm* then waits for the program to send mouse tracking information. *All X events are ignored until the proper escape sequence is received from the pty:* `ESC[Ps;Ps;Ps;Ps;Ps;Ps;T`. The parameters are *func*, *startx*, *starty*, *firstrow*, and *lastrow*. *func* is non-zero to initiate hilite tracking and zero to abort. *startx* and *starty* give the starting x and y location for the highlighted region. The ending location tracks the mouse, but will never be above row *firstrow* and will always be above row *lastrow*. (The top of the screen is row 1.) When the button is released, *xterm* reports the ending position one of two ways: if the start and end coordinates are valid text locations: `ESC[It CxCy`. If either coordinate is past the end of the line: `ESC[IT CxCyCxCyCxCy`. The parameters are *startx*, *starty*, *endx*, *endy*, *mousex*, and *mousey*. *startx*, *starty*, *endx*, and *endy* give the starting and ending character positions of the region. *mousex* and *mousey* give the location of the mouse at button up, which may not be over a character.

Tektronix 4014 Mode

Most of these sequences are standard Tektronix 4014 control sequences. The major features missing are the write-thru and defocused modes. This document does not describe the commands used in the various Tektronix plotting modes, but does describe the commands to switch modes.

<code>BEL</code>	Bell (Ctrl-G)
<code>BS</code>	Backspace (Ctrl-H)
<code>TAB</code>	Horizontal Tab (Ctrl-I)
<code>LF</code>	Line Feed or New Line (Ctrl-J)
<code>VT</code>	Vertical Tab (Ctrl-K)
<code>FF</code>	Form Feed or New Page (Ctrl-L)
<code>CR</code>	Carriage Return (Ctrl-M)
<code>ESC</code> <code>ETX</code>	Switch to VT102 Mode
<code>ESC</code> <code>ENQ</code>	Return Terminal Status
<code>ESC</code> <code>LF</code>	PAGE (Clear Screen)
<code>ESC</code> <code>ETB</code>	COPY (Save Tektronix Codes to File)
<code>ESC</code> <code>CAN</code>	Bypass Condition
<code>ESC</code> <code>SUB</code>	GIN mode
<code>ESC</code> <code>FS</code>	Special Point Plot Mode
<code>ESC</code> <code>GS</code>	Graph Mode (same as <code>GS</code>)
<code>ESC</code> <code>RS</code>	Incremental Plot Mode (same as <code>RS</code>)
<code>ESC</code> <code>US</code>	Alpha Mode (same as <code>US</code>)
<code>ESC</code> <code>8</code>	Select Large Character Set
<code>ESC</code> <code>9</code>	Select #2 Character Set
<code>ESC</code> <code>:</code>	Select #3 Character Set
<code>ESC</code> <code>;</code>	Select Small Character Set
<code>ESC</code> <code>]</code> P_s <code>:</code> $P_t BEL$	Set Text Parameters $P_s = 0$ → Change Window Name and Title to P_t $P_s = 1$ → Change Icon Name to P_t $P_s = 2$ → Change Window Title to P_t $P_s = 4$ <code>6</code> → Change Log File to P_t
<code>ESC</code> <code>^</code>	Normal Z Axis and Normal (solid) Vectors
<code>ESC</code> <code>a</code>	Normal Z Axis and Dotted Line Vectors

<code>ESC b</code>	Normal Z Axis and Dot-Dashed Vectors
<code>ESC c</code>	Normal Z Axis and Short-Dashed Vectors
<code>ESC d</code>	Normal Z Axis and Long-Dashed Vectors
<code>ESC h</code>	Defocused Z Axis and Normal (solid) Vectors
<code>ESC i</code>	Defocused Z Axis and Dotted Line Vectors
<code>ESC j</code>	Defocused Z Axis and Dot-Dashed Vectors
<code>ESC k</code>	Defocused Z Axis and Short-Dashed Vectors
<code>ESC l</code>	Defocused Z Axis and Long-Dashed Vectors
<code>ESC p</code>	Write-Thru Mode and Normal (solid) Vectors
<code>ESC q</code>	Write-Thru Mode and Dotted Line Vectors
<code>ESC r</code>	Write-Thru Mode and Dot-Dashed Vectors
<code>ESC s</code>	Write-Thru Mode and Short-Dashed Vectors
<code>ESC t</code>	Write-Thru Mode and Long-Dashed Vectors
<code>FS</code>	Point Plot Mode
<code>GS</code>	Graph Mode
<code>RS</code>	Incremental Plot Mode
<code>US</code>	Alpha Mode

F

Translation Table Syntax

This appendix describes the basic syntax of translation table resources, described in Chapter 11, Setting Resources.

In This Appendix:

Event Types and Modifiers	819
Detail Field	821
Modifiers	822
Complex Translation Examples	822

F

Translation Table Syntax

This appendix explains some of the more complex aspects of translation table syntax. It probably gives more detail than the average user will need but we've included it to help clarify this rather complicated topic.

Event Types and Modifiers

The syntax of the translation table is sufficiently general to encompass a wide variety of events and circumstances. Event translations can be specified to handle characteristic user interface idioms like double clicking, dragging, or combining keyboard modifiers with pointer button input. To specify translations that use these features, it is necessary to learn more about the detailed syntax used to specify translations.

An activity susceptible to translation is a sequence of events and modifiers (that perform an action). Events are specified in angle brackets and modifiers precede the event they modify. The legal events that can be specified in a translation are as shown in Table F-1.

Table F-1. Event Types and Their Abbreviations

Event Name	Event Type	Abbreviations/Synonyms
KeyPress	Keyboard	Key, KeyDown
KeyUp	Keyboard	KeyRelease
ButtonPress	Mouse Button	BtnDown
ButtonRelease	Mouse Button	BtnUp
Btn1Down	Mouse Button Press	
.		
.		
Btn5Down		
Btn1Up	Mouse Button Release	
.		
.		
Btn5Up		

Table F-1. Event Types and Their Abbreviations (continued)

Event Name	Event Type	Abbreviations/Synonyms
MotionNotify	Mouse Motion	Motion, MouseMoved, PtrMoved
ButtonMotion	Motion w/any Button Down	BtnMotion
Button1Motion	Motion w/Button Down	Btn1Motion
.		.
.		.
Button5Motion		Btn5Motion
EnterNotify	Mouse in Window	Enter, EnterWindow
LeaveNotify		LeaveWindow, Leave
FocusIn	Keyboard Input Focus	
FocusOut		
KeymapNotify	Changed Key Map	Keymap
ColormapNotify	Changed Color Map	Clrmap
Expose	Related Exposure Events	
GraphicsExpose		GrExp
NoExpose		NoExp
VisibilityNotify		Visible
CreateNotify	Window Management	Create
DestroyNotify		Destroy
UnmapNotify		Unmap
MapNotify		Map
MapRequest		MapReq
ReparentNotify		Reparent
ConfigureNotify		Configure
ConfigureRequest		ConfigureReq
GravityNotify		Grav
ResizeRequest		ResReq
CirculateNotify		Circ
CirculateRequest		CircReq
PropertyNotify		Prop
SelectionClear	Intra-client Selection	SelClr
SelectionRequest		SelReq
SelectionNotify		Select

The possible modifiers of an event are listed in the table. The modifiers Mod1 through Mod5 are highly system-dependent and may not be implemented by all servers.

Table F-2. Key Modifiers

Event Modifiers	Abbreviation
Ctrl	c
Meta	m
Shift	s
Lock	l
Any	
ANY	
None	
Mod1	1
.	.
.	.
Mod5	5

Detail Field

To provide finer control over the translation process, the event part of the translation can include an additional “detail.” For example, if you want the event to require an additional keystroke, for instance, an A key, or a Control-T, then that keystroke can be specified as a translation detail. The default detail field is *ANY*.

The valid translation details are event-dependent. For example, to specify the above example for keypress events, you would use:

```
<Key>A
```

and:

```
Ctrl<Key>T
```

respectively.

Key fields can be specified by the keysym value, as well as by the keysym symbolic name. For example, the keysym value of the Delete key is 0xffff. Keysym values can be determined by examining the file `<X11/keysymdef.h>` or by using the *xmodmap* client. (See Chapter 14, *Setup Clients*, for information about *xmodmap*.) Unfortunately, with some translations the keysym value may actually be required, since not all keysym symbolic names may be properly interpreted.

Modifiers

Modifiers can be closely controlled to define exactly which events can be specified. For example, if you want the action to be performed by pointer button clicks but not by pointer button clicks with the Control or Shift key down, these limitations can be specified. Similarly, if you don't care if there are modifiers present, this can also be specified.

Table F-3 lists the available event modifiers.

Table F-3. Event Modifiers and Their Meanings

Modifier	Meaning
None <i><event></i>	No modifiers allowed.
<i><event></i>	Doesn't care. Any modifiers okay.
Mod1 Mod	Mod1 and Mod2, plus any others (i.e., anything that includes m1 and m2).
!Mod1 Mod2 <i><event></i>	Mod1 and Mod2 but nothing else.
Mod1 ~Mod2 <i><event></i>	Mod1 and not Mod2.

Complex Translation Examples

The following translation specifies that function *f* is to be invoked when both the Shift key and the third pointer button are pressed.

```
Shift<Btn3Down>: f()
```

To specify that both the Control and Shift keys are to be pressed use:

```
Ctrl Shift<Btn3Down>: f()
```

To specify an optional repeat count for an activity, put a number in parentheses after the action. The number refers to the whole translation. To make the last example require a double-click, with both Control and Shift keys pressed, use:

```
Ctrl Shift<Btn3Down>(2): f()
```

The server distinguishes between single-clicks and double-clicks based on a pre-programmed timing interval. If a second click occurs before the interval expires, then the event is interpreted as a double-click; otherwise the event is interpreted as two single-clicks. The variable `clickTime` is maintained deep in the internals of X. Unfortunately, thus far there is no way to set this time interval to match user preference. Currently it is set to be 200 milliseconds.

A translation involving two or more clicks can be specified as (2+) in the previous example. In general, a plus sign following the number *n* would mean *n* or more occurrences of the event.

Multiple events can be specified by separating them with commas on the translation line. To indicate pressing button 1, pressing button 2, then releasing button 1, and finally releasing button 2, use:

```
<Btn1Down>, <Btn2Down>, <Btn1Up>, <Btn2Up>: f()
```

Another way to describe this action in English would be to say “while button 1 is down, click button 2.” “Meaningless” pointer movement is generally ignored. In the previous case, for example, if pointer motion occurred while the buttons were down, it would not interfere with detection of the event. Thus, inadvertent pointer jiggling will not thwart even the most complex user-input sequences.

G

Widget Resources

In addition to application-specific resources, you can specify resources for an application's component widgets. This appendix provides a brief overview of how widgets are used in X Toolkit programs. It then describes each of the Athena and Motif widgets, noting those widget resources a user can specify.

In This Appendix:

The Widget Class Hierarchy	827
Widgets in the Application	830
What all this Means	832
Complications	833
Athena Widget Resources	834
Box	836
Command	836
Dialog	837
Form	837
Grip	839
Label	839
List	840
MenuBar	841
Paned	842
Panner	844
Porthole	845
Repeater	846
Scrollbar	846
Simple	848
SimpleMenu	848
Sme	849
SmeBSB	850
SmeLine	851



StripChart	851
Text	852
Toggle	854
Tree	855
Viewport	856
Motif Widget Resources	857
ArrowButton	858
BulletinBoard	858
CascadeButton	859
Command	860
DialogShell	861
Display	861
DrawingArea	862
DrawnButton	863
FileSelectionBox	863
Form	864
Frame	866
Gadget	867
Label	868
List	869
MainWindow	871
Manager	871
MenuShell	873
MessageBox	873
PanedWindow	874
Primitive	876
PushButton	877
RowColumn	878
Scale	881
Screen	883
ScrollBar	884
ScrolledWindow	885
SelectionBox	886
Separator	888
Text	888
TextField	891
ToggleButton	892
VendorShell	894

Widget Resources

As suggested on the reference pages for various clients, you can set not only resources defined by the application itself, but also resources that apply to any of the widgets that make up the application. The reference page for the application sometimes lists the most important of these resources, but for fuller customization, you need to know more about each widget.

Unfortunately, the design of the X Toolkit is such that to really do the right thing, you probably need to know a bit more about Toolkit programming than the average user might like.

In this appendix, we provide both some introductory concepts about how widgets are used in X Toolkit programs, and reference information about each class of widgets. If you are a Toolkit programmer or other sophisticated user, feel free to skip right to the widget reference section later in this appendix.

The Widget Class Hierarchy

The first thing you need to know is how widgets are built.

Rather than starting each widget from scratch, the widget programmer starts with a copy of another, more basic widget, and modifies it. This process is called *subclassing* the widget, and the sequence of widgets leading up to the one you see is called its *class hierarchy*. Because of the way subclassing works, a widget *inherits* all of the characteristics of its superclasses, except those that are explicitly overridden or changed.

The class hierarchy starts with a class called Object, which defines some basic characteristics common to all widgets—namely the ability to understand resources, and to be linked to applications via a mechanism referred to as a callback. When you click on a “quit” button, and the application quits, that is because the widget has communicated with the application via a callback.

RectObj is a subclass of Object. RectObj adds various resources having to do with the fact that widgets are rectangular: width, height, borderWidth, and x, y positions. RectObj also adds resources for sensitivity—the fact that a widget can be temporarily “disabled” by a client. For example, an application might disable a menu item that closes a file if no file is open.

Core is the first true widget in the class hierarchy. Object and RectObj don't have windows associated with them, and can never be "instantiated"—created and mapped to the screen. In fact, prior to Release 4, they were "invisible" even to Toolkit programmers, who simply assumed that Core was the root of the widget hierarchy.

The reason we now talk about Object and RectObj is that since R4, the Toolkit supports a different class of object, known colloquially as a gadget, which is subclassed directly from RectObj, and does not have a window associated with it. It can be used only within a widget that understands how to manage gadgets, and allocates some of its own window space to display them. This is typically done when there are many identical widgets. The only gadgets in the Athena widget set are the SmeBSB and SmeLine gadgets used to implement panes in a SimpleMenu widget. Motif offers its programmers both widget and gadget versions of many of its objects, including all kinds of command buttons.

At any rate, for most purposes, you can still act as though Core is the root of the widget hierarchy, since all widgets are subclassed from it, and therefore share all of its resources. The phrase "Core resources" is a fluke of terminology that can be misleading to new users. Because it sounds meaningful just as a general term, it isn't clear that the Core resources are really the resources of a particular widget class (rather than something magically recognized as central or "core" by the X Toolkit.)

Let's take a brief look at some of the Core resources, which appear in Table G-1. The list includes resources inherited from Object and RectObj, plus those added by Core.

Table G-1. Core Resources

Name	Class
background	Background
borderColor	BorderColor
borderWidth	BorderWidth
height	Height
width	Width
x	Position0
y	Position0

Some of these Core resources set obvious characteristics of a widget: background (color), borderColor, and borderWidth (in pixels). height and width specify the dimensions of the widget in pixels. x and y represent the x,y coordinates of the widget in relation to its parent.

Note that Table G-1 isn't actually a complete list of all of the Core resources, but only of those that might be set by users. Some resources (such as callbacks) can only be set by programmers. The Toolkit doesn't even support a mechanism for understanding how to set them in a resource file.

In addition, font and foreground are two resources that are so common that you might expect them to be Core resources, but they are not. They are defined individually by each of the widget classes that use them. This can be confusing, especially since they do correspond

not iconified. A `TopLevelShell` is used by an application that has more than one completely independent window, as the class for its secondary “top level” windows.

For all practical purposes, you don’t need this much information about shell widgets. As we’ll see shortly, the only reference to a shell widget in a resource specification is typically via the application name, which the shell widget takes as its own.

Returning to widgets that you actually do see and interact with, let’s consider the class derivation of a widget like the Athena Command widget, which is used to implement buttons you can click on with the mouse to ask the application to do something.

The Athena Command widget is a subclass of the Label widget, which is a subclass of the Simple widget, which in turn is a subclass of Core. As a result, Command inherits all of the Core resources, plus the resources of the Athena Simple widget plus the resources of the Label widget—such as the ability to display a label, in a particular font. Command adds the ability (defined by the programmer, not the user) to call a particular application function when the button is clicked on. We’ll come back to the complete Athena widget hierarchy later in the widget reference section.

Widgets in the Application

Widget inheritance of resources from superclasses is an important part of the background to understanding how to affect the widget resources in the application, but it is not the whole story. Let’s talk for a moment about how these widgets are used.

To make things more concrete, let’s look at an actual application. *xclipboard* is a good choice. It uses several different widget classes, but isn’t too complex. Figure G-1 illustrates the widgets that make up *xclipboard*.

Every Toolkit application begins with a call to a function called `XtAppInitialize()`, which looks something like this:

```
top = XtAppInitialize(... , "XClipboard", ... );
```

The second argument to this function gives the class name of the application. This name becomes the start of any resource specification for the application. And we know that if *xclipboard* has an app-defaults file, it will be called *XClipboard*, since that name is taken from the class name of the application. Notice that there’s no magic here: this is under the explicit control of the application programmer. If the application doesn’t follow the conventions for the class and instance names, it needs to document the names that are used.

One of the things that `XtAppInitialize()` does is create a `ApplicationShell` widget. The variable name (before the equals sign), `top`, is the name that the programmer uses to refer to this widget whenever she needs to use it in the application. This name is completely irrelevant to the name the widget publishes for itself as its instance name.

Next, the program begins to create the widgets in the application, using a function called `XtCreateManagedWidget()`. The first widget to be created is the main application widget, which in this case is a Form widget.

```
parent = XtCreateManagedWidget("form", formWidgetClass, top, ... );
```

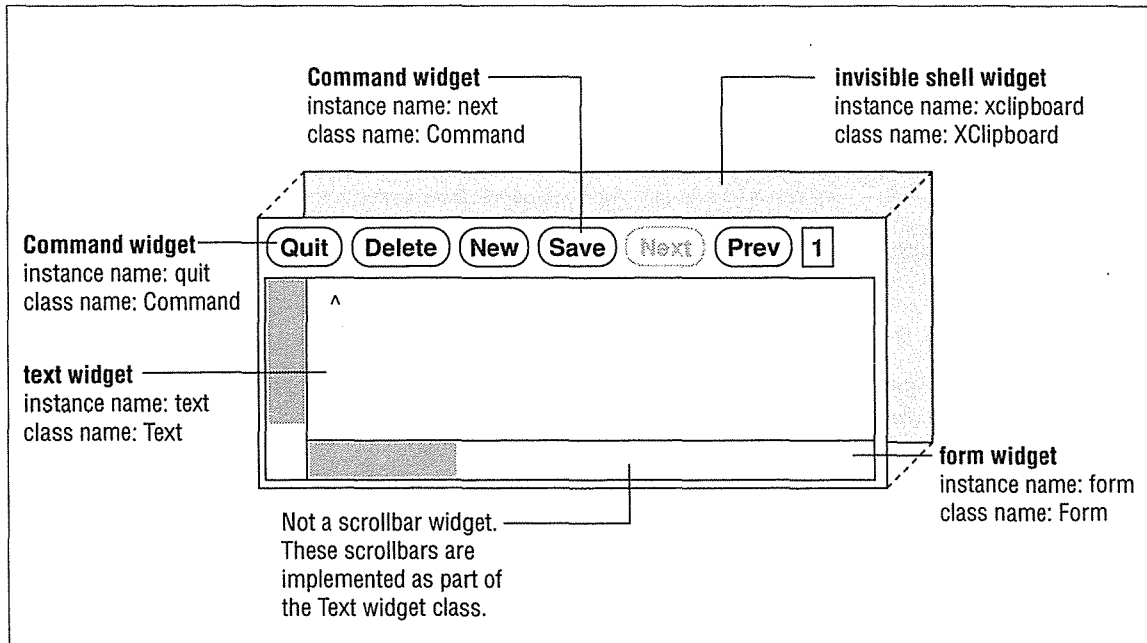


Figure G-1. Anatomy of an X Toolkit application

The first argument to `XtCreateManagedWidget()` is the instance name of the widget (`form`)—this is the name that will be used to refer to it in resource files. The second argument is a symbol identifying which widget class this widget should be.

Notice that the instance name is entirely arbitrary, and depends completely on the whim of the application programmer. Many applications that use only one instance of a widget class will give it an instance name that mirrors the class name, except in lower case, as was done here. But you can see that the programmer could just as well have given the widget the instance name “foo” or “main” or “howdy_doddy.” The implication is that unless the client’s man page documents a widget instance name, you won’t know what to use in a resource file.*

The class name, on the other hand, is a part of the definition of a widget’s class. It is always the same.

The third argument is the widget’s parent—the geometry-managing widget that this widget will be displayed inside, and which will control its size and position. Notice that the parent of the form is `top`—the shell widget created by `XtAppInitialize()`. As noted earlier, Shell widgets take just one child, and resize themselves so they fit completely behind that child, and are invisible.

Remember, though, that the program’s internal name for the shell widget is not important when it comes to resource specifications. The Shell widget takes its “resource name and class” from the `XtAppInitialize()` call.

*As of R4, most of the MIT client reference pages list the instance names of all the widgets in the application.

If you're following the flow of the argument, you can see that to refer to this widget in a resource file, you could use any of the following resource specifications:

<code>xclipboard.form</code>	<i>instance name for both the shell widget and form widget</i>
<code>XClipboard.Form</code>	<i>class name for both the shell widget and form widget</i>
<code>XClipboard.form</code>	<i>class name for the shell, and instance name for the form</i>
<code>xclipboard.Form</code>	<i>instance name for the shell, and class name for the form</i>

as well as any analogous loose bindings.

The form widget (named "parent" for internal reference within the application) is used in turn as the parent of the various command widgets and the text widget:

```
quit = XtCreateManagedWidget("quit", commandWidgetClass, parent, ... );
delete = XtCreateManagedWidget("delete", commandWidgetClass, parent, ... );
new = XtCreateManagedWidget("new", commandWidgetClass, parent, ... );
nextButton = XtCreateManagedWidget("next", commandWidgetClass, parent, ... );
prevButton = XtCreateManagedWidget("prev", commandWidgetClass, parent, ... );
text = XtCreateManagedWidget("text", textWidgetClass, parent, ... );
```

This "parent-child relationship" between Composite widgets and their children is what is expressed in the instance hierarchy of the widget. So, for example, the Command widget instance named `quit` is a child of the Form widget instance named `form`, which in turn is a child of a Shell widget, which takes as its name the application name `xclipboard`.

What All This Means

The fully-specified instance name of any widget in an application is determined by the parent-child relationships of every widget in the application. First, there is always a Shell widget, which takes as its name the application name. Then, there are one or more Composite widgets, which contain other widgets. Finally, at the end of the chain, you have a simple widget, with the resources it defines, as well as the resources it inherits from its superclasses.

Don't confuse the class names of the widgets in the instance hierarchy with the class inheritance hierarchy of each widget. Figure G-2 tries to make the relationships clear.

In Figure G-2, the `quit` widget gets its instance name from the relationship of widgets within the application. But it gets its resources from the class hierarchy of the widgets that the programmer used to develop the Command widget class.

Remember that the instance names of the widgets are completely arbitrary; even though it is not unusual to see a Form widget with the instance name `form`, there is nothing required about this. As a result, you need to look at the documentation for the application, not the widget, to find out the appropriate instance names.

The resources that a given widget class has are the result of its class inheritance hierarchy, which is defined by the widget programmer who originally designed the widget class. Thus, when you want to set resources for a widget like Command, you need to look not only in the section of this appendix that describes Command and its resources, but also the sections on each of its superclasses.

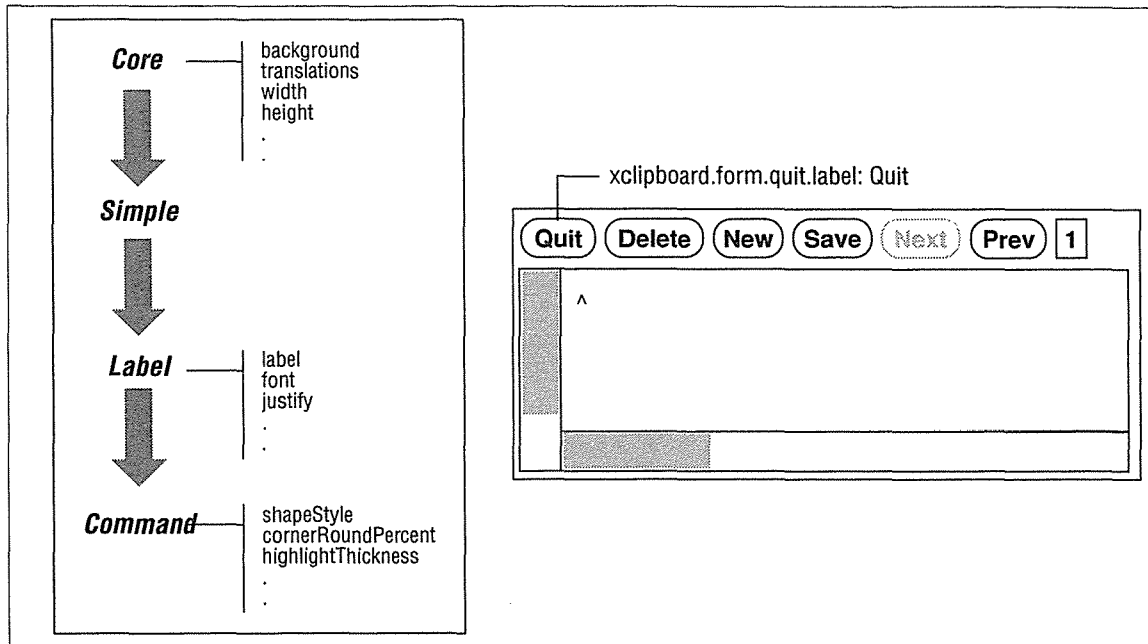


Figure G-2. Resource names and class inheritance

Complications

There are a number of provisos that modify this (hopefully by now clear and simple) picture:

- Even though a widget inherits a resource, it may not use it. For example, the Command widget class inherits the `borderWidth` resource from the Core widget class, but it does not actually use this information to redraw its border if you change the resource. A resource is just data you provide to the widget. Whether or not the widget does anything with that data is up to its designer. If you set a resource and nothing seems to happen, you might have done something wrong . . . but you might also have set the resource correctly, and the widget simply chose to ignore it.
- Even when a widget does use a resource, you can't necessarily set it from a resource file. There are two reasons for this:
 - The programmer has the option to “hardcode” the value of a resource when creating a widget. If he does this, all resource specifications for that resource are ignored.
 - Some resources are designed only for programmer use. Some of these can't ever be specified in a resource file, since the data type of the resource isn't a text string, and the Toolkit doesn't provide any automatic conversion. (Features like colors can be specified in resource files, even though a color name is not actually the color itself, because the X Toolkit automatically converts a color name to the appropriate internal format).

The following pages document only resources that are theoretically settable from resource files. (That is, if no converter exists, we've assumed that the resource is only for programmer use and have deleted it from the list.) However, there are many other

resources listed that are most likely hardcoded by the programmer. Unfortunately, there is no way to tell in advance whether they will or will not be hardcoded in a particular application.

- The “default values” listed for each widget resource may or may not apply to an actual application. These are the default values for the widget. An application can override them, either in the program code, or in an application defaults file. But inasmuch as the defaults are reasonable, they will usually be unchanged.

With this background, you’re now ready to navigate the widget reference information contained in this appendix. For each widget and gadget in the Athena and Motif widget sets, we’ve given a brief description, a summary of its class hierarchy, and a list of the new resources it defines.

Athena Widget Resources

All of the standard MIT applications described in this book have been built using the Athena widget set. Figure G-3 shows the complete class hierarchy of the Athena widgets. The widgets shown in gray are defined by the X Toolkit intrinsics, and are common to all Xt-based widget sets.

The *listres* application, without any arguments, lists the inheritance hierarchy for each of the Athena widgets. Given the name of any widget class, it lists all of the resources for that widget, and which superclass they are inherited from. For example:

```
% listres label
WidgetClass          Instance  Class          Type
-----
label:  Core\Simple\Label
Core          accelerators  Accelerators   AcceleratorTable
Core          ancestorSensitive Sensitive      Boolean
Core          background    Background     Pixel
Core          backgroundPixmap Pixmap         Pixmap
Label         bitmap       Pixmap         Bitmap
Core          borderColor  BorderColor    Pixel
Core          borderPixmap Pixmap         Pixmap
Core          borderWidth  BorderWidth    Dimension
Core          colormap    Colormap       Colormap
Simple        cursor       Cursor         Cursor
Simple        cursorName   Cursor         String
Core          depth       Depth          Int
Core          destroyCallback Callback       Callback
Label         encoding    Encoding       UnsignedChar
Label         font        Font           FontStruct
Label         foreground  Foreground     Pixel
Core          height     Height         Dimension
Simple        insensitiveBorder Insensitive    Pixmap
Label         internalHeight Height          Dimension
Label         internalWidth Width           Dimension
Label         justify    Justify        Justify
Label         label      Label          String
Label         leftBitmap  LeftBitmap     Bitmap
```

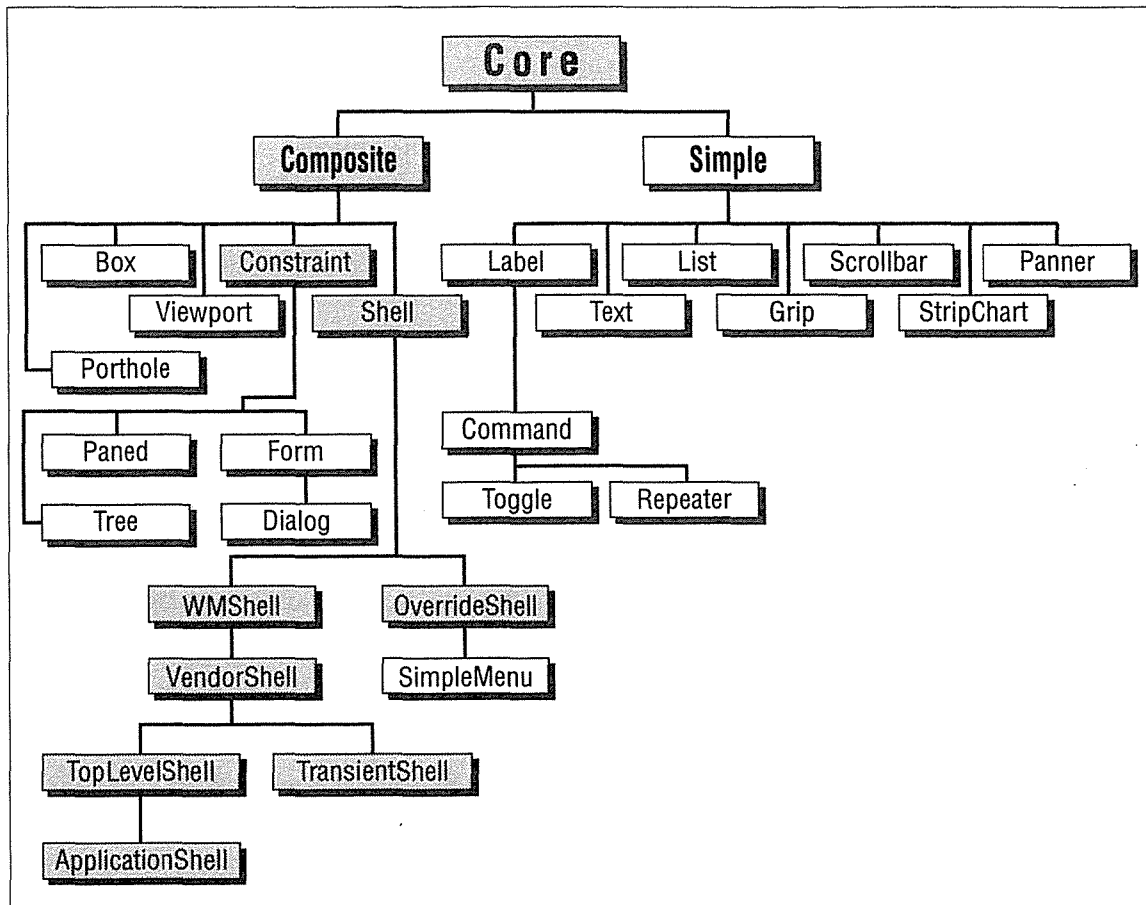


Figure G-3. Inheritance among the Athena widgets

Core	mappedWhenManaged	MappedWhenManaged	Boolean
Simple	pointerColor	Foreground	Pixel
Simple	pointerColorBackground	Background	Pixel
Label	resize	Resize	Boolean
Core	screen	Screen	Screen
Core	sensitive	Sensitive	Boolean
Core	translations	Translations	TranslationTable
Core	width	Width	Dimension
Core	x	Position	Position
Core	y	Position	Position

Not all of the resources listed by *listres* can be set in a resource file. However, this listing can provide a handy quick reference.

The rest of this section provides more detailed information on the Athena widgets and their resources. For each widget, there is a brief description and a list of the new resources defined by the widget. Note that these resource lists include only those resources that can be set in resource files; they are not complete lists. For the full reference material on the Athena widgets, see Volume Five, *X Toolkit Intrinsic Reference Manual*.

Box

The Box widget provides geometry management of arbitrary widgets in a box of a specified dimension. Box moves but does not resize its children. The children are rearranged when the Box is resized, when its children are resized, or when children are managed or unmanaged. The Box widget always attempts to pack its children as closely as possible within the geometry allowed by its parent.

Box widgets are commonly used to manage a related set of Command widgets and are frequently called ButtonBox widgets, but the children are not limited to buttons.

The children are arranged on a background that has its own specified dimensions and color.

The class hierarchy for Box is: Core → Composite → Box.

Resources

The following new resources are associated with the Box widget:

`hSpace` (class `HSpace`)

Number of pixels to the left or to the right of each child. Default is 4.

`orientation` (class `Orientation`)

Specifies whether the preferred shape of the box is tall and narrow (`vertical`, the default) or short and wide (`horizontal`).

`vSpace` (class `VSpace`)

Number of pixels above or below each child. Default is 4.

Command

The Command widget is an area, often rectangular, that contains a text or pixmap label and calls an application function when “pressed” with a pointer button. This selectable area is sometimes referred to as a “button.” When the pointer cursor is on the button, the button border is highlighted to indicate that the button is ready for selection. When a pointer button is pressed, the command widget indicates that it has been selected by reversing its foreground and background colors.

The class hierarchy for Command is: Core → Simple → Label → Command.

Resources

The following new resources are associated with the Command widget:

`highlightThickness` (class `Thickness`)

The thickness of the line drawn when the button is highlighted.

`shapeStyle` (class `ShapeStyle`)

Nonrectangular buttons may be created using this resource. Nonrectangular buttons are supported only on a server that supports the Shape Extension. If nonrectangular

buttons are specified for a server lacking this extension, the shape is ignored and the widgets will be rectangular. The following shape names are currently supported: `rectangle`, `oval`, `ellipse`, and `roundedRectangle`.

`cornerRoundPercent` (class `CornerRoundPercent`)

When a `ShapeStyle` of `roundedRectangle` is used, this resource controls the radius of the rounded corner. The radius of the rounded corners is specified as a percentage of the length of the shortest side of the widget.

Dialog

The Dialog widget prompts you for additional input. The typical Dialog widget contains three areas. The first line contains a description of the function of the Dialog widget, for example, the string *Filename:*. The second line contains an area into which you type input. The third line can contain buttons that let you confirm or cancel the Dialog input.

Dialog is not really a widget, but an interface to a widget. It might also be thought of as a compound widget. It includes a label widget, a command widget, and a text widget as components. These could theoretically appear as subwidgets in a resource specification.

The class hierarchy for Dialog is: `Core` → `Composite` → `Constraint` → `Form` → `Dialog`

Resources

The following new resources are associated with the Dialog widget:

`icon` (class `Icon`)

The name of a pixmap to be displayed immediately to the left of the Dialog widget's label.

`label` (class `Label`)

A Latin1 string to be displayed at the top of the Dialog widget.

`value` (class `Value`)

An initial value for the string field into which you will enter text. By default, no text entry field is available. Specifying an initial value for `value` activates the text entry field. If string input is desired but no initial value is to be specified, then set this resource to " " (empty string).

Form

The Form widget can contain an arbitrary number of children of any class. The Form provides geometry management for its children, including individual control of the position of each child. The initial positions of the children may be computed relative to the positions of other children. When the Form is resized, it computes new positions and sizes for its children.

The class hierarchy for Form is: Core → Composite → Constraint → Form

Resources

The following new resource is associated with the Form widget:

`defaultDistance` (class Thickness)

Specifies the default value for `horizDistance` and `vertDistance`. This value is four pixels, by default. The default width of the Form is the minimum width needed to enclose the children after computing their initial layout, with a margin of `defaultDistance` at the right and bottom edges. If a width and height is assigned to the Form that is too small for the layout, the children will be clipped by the right and bottom edges of the Form.

Form is a subclass of Constraint, which means it has a special kind of resources called constraint resources. These resources apply to—and are specified as if they belong to—the child of the Form rather than to the Form itself. For example, *xcalc* uses a Form widget to organize its buttons. The resources below apply to the buttons, rather than to the Form (e.g., `xcalc.ti.button11.horizDistance : 4`). Form specifies the following constraint resources for its children:

`bottom` (class Edge)

`top` (class Edge)

`left` (class Edge)

`right` (class Edge)

Specify how to reposition the bottom, top, left, and right, respectively, of a child widget when the Form is resized. These resources can take one of five values. The values `ChainTop`, `ChainBottom`, `ChainLeft`, and `ChainRight` maintain a constant distance from an edge of the child to the top, bottom, left, and right edges, respectively, of the Form. The value `Rubber` (default) maintains a proportional distance from the edge of the child to the left or top edge of the Form. The proportion is determined from the initial position of the child and the initial size of the Form.

`fromHoriz` (class Widget)

`horizDistance` (class Thickness)

Specify a child widget's horizontal position relative to another widget within the Form. `fromHoriz` is the name of the widget relative to which the child widget is placed, and `horizDistance` is the number of pixels separating the two widgets. For example, if `horizDistance` is 10, the child widget will be placed 10 pixels to the right of the widget defined in `fromHoriz`. If `fromHoriz` is not defined, then `horizDistance` is measured from the left edge of the Form.

`fromVert` (class Widget)

`vertDistance` (class Thickness)

Similar to previous resources, except that `fromVert` and `vertDistance` position a child widget by a specified number of pixels *vertically* away from a specified widget. If no widget is specified for `fromVert`, then `vertDistance` is measured from the top of the Form.

resizable (class Boolean)

Specifies whether children are allowed to resize themselves. Default is False.

Grip

The Grip widget provides a small region that accepts button presses and button releases. The Grip widget is typically used as an attachment point for visually repositioning an object (for example, the pane border in a Paned widget).

The class hierarchy for Grip is: Core → Simple → Grip

Resources

Grip does not have any new user-settable resources associated with it. The following Core resources may be useful with the Grip widget: foreground, width, height, border-width.

Label

A Label is a non-editable text string or pixmap that is displayed within a window. The string may contain multiple lines of Latin1 characters. It can be aligned to the left, right, or center of its window. A Label can be neither selected nor directly edited by the user.

The class hierarchy for Label is: Core → Simple → Label

Resources

The following new resources are associated with the Label widget:

bitmap (class Pixmap)

Specifies a bitmap to display in place of the text label. In a resource file, the resource should be specified as the name of a file in the bitmap utility format that is to be loaded into a pixmap. The string can be an absolute or a relative filename. If a relative filename is used, the directory specified by the resource name `bitmapFilePath` or the resource class `BitmapFilePath` is added to the beginning of the specified filename. If the `bitmapFilePath` resource is not defined, the default directory on a POSIX-based system is `/usr/include/X11/bitmaps`.

encoding (class Encoding)

Specifies whether the widget uses 8-bit or 16-bit text functions. New in R5.

font (class font)

The font of the label.

foreground (class Foreground)

The color of the text string or pixmap.

`internalHeight` (class `Height`)

Represents the distance in pixels between the top and bottom of the label text or bitmap and the horizontal edges of the Label widget. Default is 2 pixels.

`internalWidth` (class `Width`)

Represents the distance in pixels between the ends of the label text or bitmap and the vertical edges of the Label widget. Default is 4 pixels.

`justify` (class `Justify`)

Specifies left, center, or right alignment of the label string within the Label widget. One of the values `left`, `center`, or `right` can be specified.

`label` (class `Label`)

Specifies the text string that is to be displayed in the button if no bitmap is specified. Note that the label may be hardcoded by the application.

`leftBitmap` (class `LeftBitmap`)

Specifies the name of a bitmap to display in the left margin of the Label. All 1's in the bitmap are rendered in the foreground color and all 0's will be drawn in the background color. New in R5.

`resize` (class `Resize`)

A Boolean value that specifies whether the Label widget should attempt to resize to its preferred dimensions whenever `XtSetValues` is called for it. Default is `True`. Not usually set by users.

List

The List widget is a rectangle that contains a list of text strings formatted into rows and columns. When one of the strings is selected, it is highlighted, and an application callback routine is invoked. Only one string may be selected at a time. Note that most of the List resources are for application use.

The class hierarchy for List is: `Core` → `Simple` → `List`

Resources

The following new resources are associated with the List widget:

`columnSpacing` (class `Spacing`)

`rowSpacing` (class `Spacing`)

Specify the amount of space in pixels between each of the columns and rows in the list. The defaults are 6 pixels between columns and 4 pixels between rows.

`defaultColumns` (class `Columns`)

Specifies the default number of columns, which is used when neither the width nor the height of the List widget is specified or when `forceColumns` is `True`. The default is 2.

- `forceColumns` (class `Columns`)
Specifies that the default number of columns is to be used no matter what the current size of the List widget is. The default is `False`.
- `font` (class `Font`)
Specifies the font to be used to display the list.
- `foreground` (class `Foreground`)
Specifies the color to be used to paint the text of the list elements.
- `internalHeight` (class `Height`)
Represents a margin, in pixels, between the top and bottom of the list and the edges of the List widget. Default is 2 pixels.
- `internalWidth` (class `Width`)
Represents a margin, in pixels, between the left and right edges of the list and the edges of the List widget. Default is 4 pixels.
- `longest` (class `Longest`)
Specifies the length, in pixels, of the longest string in the current list. If the client knows the length, it should specify it; otherwise, the List widget computes a default length by searching through the list. This value is not typically set in resource files.
- `numberStrings` (class `NumberStrings`)
Specifies the number of strings in the current list. If a value is not specified, the list must be `NULL`-terminated. This value is not typically set in resource files.
- `pasteBuffer` (class `Boolean`)
If this is `True`, then the value of the string selected will be put into X cut buffer 0. The default is `False`. (Normally, the selected item is simply passed to the application. For example, a filename might be passed to the application's "open" routine.)
- `verticalList` (class `Boolean`)
If this is `True`, the elements in the list are arranged vertically; if `False`, the elements are arranged horizontally.

MenuButton

The `MenuButton` widget is a subclass of the `Command` widget that is used to pop-up a menu. It is an area, often rectangular, that contains a text or pixmap label. This selectable area is referred to as a button. When the pointer cursor is on the button, the button border is highlighted to indicate that the button is ready for selection. When pointer button 1 is pressed, the `MenuButton` widget pops up the menu that has been named in the `menuName` resource.

The class hierarchy for `MenuButton` is: `Core` → `Simple` → `Label` → `Command` → `MenuButton`

Resources

MenuButton has the `menuName` resource associated with it, but this resource can only be set in application code.

Paned

The Paned widget manages children in a vertically or horizontally tiled fashion. You may resize these panes by using the *grips* that appear near the right or bottom edge of the border between two panes.

When you position the pointer on a grip, pressing the pointer button will display an arrow that indicates which pane is being resized. By keeping the pointer button down, you can move the pointer up and down (or left and right). This, in turn, changes the border between the panes, causing one pane to shrink and some other pane (or panes) to grow. The size of the Paned widget will not change.

The choice of panes that are resized is a function of the `min`, `max`, and `skipAdjust` constraints on the other panes. With the default bindings, button 1 resizes the pane above or to the left of the selected grip, button 3 resizes the pane below or to the right of the selected grip, and button 2 repositions the border between two panes only.

The class hierarchy for Paned is: Core → Composite → Constraint → Paned

Resources

The following new resources are associated with the Paned widget:

`betweenCursor` (class `Cursor`)

Cursor that is displayed when you are changing the boundary between two panes.

`cursor` (class `Cursor`)

Pointer cursor image that is displayed whenever the pointer is in this widget but not in any of its children (children may also inherit this cursor).

`gripCursor` (class `Cursor`)

Cursor that is displayed for the grip when it is not active.

`gripIndent` (class `GripIndent`)

Offset of grip from margin (in pixels). Default is 16.

`gripTranslations` (class `Translations`)

Button bindings for the grip.

`horizontalBetweenCursor` (class `Cursor`)

Cursor that is displayed for the grip when you are changing the boundary between two horizontal panes. Default is `sb_up_arrow`.

`horizontalGripCursor` (class `Cursor`)

Cursor that is displayed for the grips in a horizontal Paned widget when they are not active. Default is `sb_h_double_arrow`.

`internalBorderColor` (class `BorderColor`)
 Internal border color of the widget's window.

`internalBorderWidth` (class `BorderWidth`)
 Amount of space (in pixels) kept between panes. Default is 1.

`leftCursor` (class `Cursor`)
 Cursor used when resizing the pane to the left of the grip. Default is `sb_left_arrow`.

`lowerCursor` (class `Cursor`)
 Cursor used when resizing the pane below the grip. Default is `sb_down_arrow`.

`orientation` (class `Orientation`)
 Orientation to use in stacking the panes. This value can be either `vertical` (the default) or `horizontal`.

`refigureMode` (class `Boolean`)
 A Boolean that specifies whether the Paned widget should adjust its children. Default is `True`.

`rightCursor` (class `Cursor`)
 Cursor used when resizing the pane to the right of the grip. Default is `sb_right_arrow`.

`upperCursor` (class `Cursor`)
 Cursor used when resizing the pane above the grip. Default is `sb_up_arrow`.

`verticalBetweenCursor` (class `Cursor`)
 Cursor that is displayed for the grip when you are changing the boundary between two vertical panes. Default is `sb_left_arrow`.

`verticalGripCursor` (class `Cursor`)
 Cursor that is displayed for the grips in a vertical Paned widget when they are not active. Default is `sb_v_double_arrow`.

Paned is a subclass of `Constraint`, which means it has special kind of resources called constraint resources. These resources apply to—and are specified as if they belong to—the child of the Paned widget rather than to the Paned widget itself. Paned specifies the following constraint resources for its children:

`allowResize` (class `Boolean`)
 A Boolean that specifies whether to accept the child's request to resize. The default, `False`, is to ignore such requests.

`max` (class `Max`)
 Maximum height for the pane (in pixels). Default is to allow unlimited height.

`min` (class `Min`)
 Minimum height for the pane (in pixels). Default is 1.

`preferredPaneSize` (class `PreferredPaneSize`)
 Preferred size of the pane.

`resizeToPreferred` (class `Boolean`)

A `Boolean` that specifies whether to resize the pane to its preferred size when the `Paned` widget is resized. Default is `False`.

`showGrip` (class `ShowGrip`)

A `Boolean` that specifies whether to show a grip for the pane. Default is `True`.

`skipAdjust` (class `Boolean`)

Specifies whether the `Paned` widget will automatically resize the pane. The default is `False`, which means that the `Paned` widget will resize the pane automatically whenever necessary. If the resource is `True`, the `Paned` widget will skip the adjustment of the pane.

Panner

The `Panner` widget is conceptually a two-dimensional scrollbar. It displays a rectangle within a rectangle—the inner rectangle (the “slider”) represents the visible portion of a larger area (the “canvas”) represented by the outer rectangle. The size of the inner rectangle represents the size of the visible area relative to the whole, and its position indicates the relative position of the visible area within the whole. You may drag the inner rectangle with the mouse (or use keyboard arrow keys) to pan through the large diagram or document (or whatever) that is being displayed. The `Panner` widget is typically used with a `Porthole` widget to scroll a third widget in two dimensions.

The `Panner` widget is a new Athena widget in R5.

The class hierarchy for `Panner` is: `Core` → `Simple` → `Panner`

Resources

The following new resources are associated with the `Panner` widget:

`allowOff` (class `AllowOff`)

Whether to allow the edges of the slider to go off the edges of the canvas. The default is `False`.

`backgroundStipple` (class `BackgroundStipple`)

The name of a bitmap pattern to be used as the background for the area representing the canvas.

`canvasHeight` (class `CanvasHeight`)

The height of the canvas.

`canvasWidth` (class `CanvasWidth`)

The width of the canvas.

`defaultScale` (class `DefaultScale`)

The percentage size that the `Panner` widget should have relative to the size of the canvas. Default is 8.

foreground (class `Foreground`)

The slider foreground color.

internalSpace (class `InternalSpace`)

The width of the internal border in pixels between a slider representing the full size of the canvas and the edge of the Panner widget. Default is 4.

lineWidth (class `LineWidth`)

The width of the lines in the rubberbanding rectangle when rubberbanding is in effect instead of continuous scrolling. The default is 0.

resize (class `Resize`)

Whether or not to resize the Panner whenever the canvas size is changed so that the `defaultScale` is maintained. Default is `True`.

rubberBand (class `RubberBand`)

Whether or not scrolling should be discrete (only moving a rubberbanded rectangle until the scrolling is done) or continuous (moving the slider itself). Default is `False`, which means that the slider is moved.

shadowColor (class `shadowColor`)

The color of the shadow underneath the slider.

shadowThickness (class `ShadowThickness`)

The width of the shadow underneath the slider.

sliderX (class `SliderX`)

sliderY (class `SliderY`)

The X and Y locations of the slider in the coordinates of the canvas.

sliderHeight (class `sliderHeight`)

sliderWidth (class `sliderWidth`)

The height and width of the slider.

Porthole

The Porthole widget provides geometry management of a list of arbitrary widgets, only one of which may be managed at any particular time. The managed child widget is reparented within the porthole and is moved around by the application (typically under the control of a Panner widget). The Porthole widget allows its managed child to request any size that is as large or larger than the Porthole itself and any location so long as the child still obscures all of the Porthole.

The Porthole widget is a new Athena widget in R5.

The class hierarchy for Porthole is: `Core` → `Composite` → `Porthole`

Resources

The Porthole widget does not have any new user-settable resources associated with it.

Repeater

The Repeater widget is a version of the Command button that triggers at an increasing rate while it is held down. It is typically used to implement valuators or certain types of scrollbars.

The Repeater widget is a new Athena widget in R5.

The class hierarchy for Repeater is: Core → Simple → Label → Command → Repeater

Resources

The following new resources are associated with the Repeater widget:

`decay` (class `Delay`)

The number of milliseconds to subtract from the repeat interval after each repetition. The interval starts at `repeatDelay` and decreases to `minimumDelay`. The default is 5 milliseconds.

`flash` (class `Boolean`)

Whether or not to flash the Repeater button whenever the timer goes off. The default is `False`.

`initialDelay` (class `Delay`)

The number of milliseconds before the Repeater widget begins to repeat. The default is 200.

`minimumDelay` (class `MinimumDelay`)

The minimum time between callbacks in milliseconds. The default is 10.

`repeatDelay` (class `Delay`)

The number of milliseconds between repetitions, once the `initialDelay` has elapsed and the widget has begun to repeat. The actual delay interval will have `decay` milliseconds subtracted from it at each repetition until it reaches `minimumDelay`.

Scrollbar

The Scrollbar widget is a rectangular area that contains a slide region and a thumb (slide bar). A Scrollbar can be used alone (to provide a graduated scale) or within a composite widget (for example, a Viewport). A Scrollbar can be aligned either vertically or horizontally.

When a Scrollbar is created, it is drawn with the thumb in a contrasting color. The thumb is normally used to scroll client data and to give visual feedback on the percentage of the client data that is visible.

The class hierarchy for Scrollbar is: Core → Simple → Scrollbar

Resources

You can set the dimensions of the Scrollbar two ways:

- By using the `width` and `height` resources, as you can for all widgets.
- By using the Scrollbar resources `length` and `thickness`, which are independent of the vertical or horizontal orientation.

The following new resources are associated with the Scrollbar widget:

`foreground` (class `Foreground`)

The color used to draw the thumb.

`length` (class `Length`)

Specifies the height for a vertical Scrollbar and the width for a horizontal Scrollbar. Default is 1 (pixel).

`minimumThumb` (class `MinimumThumb`)

Smallest size, in pixels, to which the thumb can shrink. Default is 7.

`orientation` (class `Orientation`)

Orientation of scrollbar. This value can be either `vertical` (the default) or `horizontal`. Not usually set in resource files.

`scrollDCursor` (class `Cursor`)

The cursor used for scrolling backward in a vertical Scrollbar. Default is `sb_down_arrow`.

`scrollHCursor` (class `Cursor`)

The cursor used when a horizontal Scrollbar is inactive. Default is `sb_h_double_arrow`.

`scrollLCursor` (class `Cursor`)

The cursor used for scrolling forward in a horizontal Scrollbar. Default is `sb_left_arrow`.

`scrollRCursor` (class `Cursor`)

The cursor used for scrolling backward in a horizontal Scrollbar. Default is `sb_right_arrow`.

`scrollUCursor` (class `Cursor`)

The cursor used for scrolling forward in a vertical Scrollbar. Default is `sb_up_arrow`.

`scrollVCursor` (class `Cursor`)

The cursor used when a vertical Scrollbar is inactive. Default is `sb_v_double_arrow`.

`shown` (class `Shown`)

The size of the thumb, as a percentage of the length of the Scrollbar. Default is 0.0.

`thickness` (class `Thickness`)

Specifies the width for a vertical Scrollbar and the height for a horizontal Scrollbar. Default is 14 (pixels).

`thumb` (class `Thumb`)

The pixmap used to stipple the thumb. Default is None.

`topOfThumb` (class `TopOfThumb`)

The location of the top of the thumb, as a percentage of the length of the Scrollbar.

Simple

The Simple widget defines characteristics that are inherited by non-composite widgets such as Labels, Lists, and Scrollbars. The Simple widget never appears in applications, but it does define resources that are inherited by its subclasses.

The class hierarchy for Simple is: Core → Simple

Resources

The following new resources are associated with the Simple widget:

`cursor` (class `Cursor`)

The cursor to use within the widget. Default is none.

`cursorName` (class `Cursor`)

Specifies a cursor by name from the standard cursor font to be used in the widget's window. New in R5.

`insensitiveBorder` (class `Insensitive`)

The pixmap to use to indicate that the Simple widget cannot receive input. Default is `GrayPixmap`.

`pointerColor` (class `Foreground`)

Specifies a foreground color used when creating the cursor specified in `cursorName`. New in R5.

`pointerColorBackground` (class `Background`)

Specifies a background color used when creating the cursor specified in `cursorName`. New in R5.

SimpleMenu

The SimpleMenu widget is a container for menu entries. It is a direct subclass of Shell. This is the only part of the menu that actually contains a window, since each menu pane is a gadget (a widget without a window). SimpleMenu “glues” the individual menu entries together into one menu.

The class hierarchy for SimpleMenu is: Core → Composite → Shell → OverrideShell → SimpleMenu

Resources

The following new resources are associated with the SimpleMenu widget:

`bottomMargin` (class `VerticalMargins`)

`topMargin` (class `VerticalMargins`)

The amount of space between the top or bottom of the menu and the menu entry closest to that edge. Default is 0.

`cursor` (class `Cursor`)

The shape of the mouse pointer whenever it is in this widget.

`label` (class `Label`)

This label will be placed at the top of the SimpleMenu and cannot be highlighted. The name of the label object is `menuLabel`, and it is of the class specified by the `Label-Class` resource. Using this name, it is possible to modify the label's attributes through the resource database. When the label is created, its `label` resource is hard-coded to the value of `label` and `justify` is hard-coded as `center`.

`labelClass` (class `LabelClass`)

Specifies the type of `Sme` object created as the menu label. Possibilities are `Sme`, `SmeBSB`, or `SmeLine`.

`popupOnEntry` (class `PopupOnEntry`)

The `XawPositionSimpleMenu` action pops up the SimpleMenu with its label (or first entry) directly under the pointer, by default. To pop up the menu under another entry, the application can set this resource to the menu entry that *should* be under the pointer when the menu is popped up. This allows the application to offer the user a default menu entry that can be selected without moving the pointer. Not usually settable by the user.

`rowHeight` (class `RowHeight`)

If this resource is 0 (the default), then each menu entry is given its desired height. If this resource has any other value, then all menu entries are forced to be `rowHeight` pixels high.

Sme

The `Sme` object is the base class for all menu entries that are children of SimpleMenu. While this object is intended mainly to be subclassed, it may be used in a menu to add blank space between menu entries.

The class hierarchy for `Sme` is: Object → `RectObj` → `Sme`

Resources

The Sme object does not have any new user-settable resources associated with it.

SmeBSB

The SmeBSB object is used to create a menu entry that contains a string and optional bitmaps in its left and right margins. The parent is expected to be a SimpleMenu. Since each menu entry is an independent object, the application is able to change the font, color, height, and other attributes of the menu entries on an entry-by-entry basis.

The class hierarchy for SmeBSB is: Object → RectObj → Sme → SmeBSB

Resources

The following new resources are associated with the SmeBSB object:

font (class Font)

Specifies the font used by the menu entry.

foreground (class Foreground)

Specifies the foreground color of the menu entry's window. This color is also used to render all 1's in leftBitmap and rightBitmap.

justify (class Justify)

Specifies how the label is to be rendered between the left and right margins when the space is wider than the actual text. When specifying the justification from a resource file, the values left, center, or right may be used.

label (class Label)

Specifies the string to be displayed in the menu entry. The exact location of this string within the bounds of the menu entry is controlled by the resources leftMargin, rightMargin, vertSpace, and justify.

leftBitmap (class LeftBitmap)

rightBitmap (class RightBitmap)

Specifies a name of a bitmap to display in the left or right margin of the menu entry. All 1's in the bitmap are rendered in the foreground color of the entry and all 0's will be drawn in the background color of the SimpleMenu widget. The menu entry needs to be tall enough and the appropriate margin needs to be wide enough to accept the bitmap. If care is not taken, the bitmap might extend into either another menu entry or this entry's label.

leftMargin (class HorizontalMargins)

rightMargin (class HorizontalMargins)

Specifies the amount of space (in pixels) to leave between the edge of the menu entry and the label string.

`vertSpace` (class `VertSpace`)

Specifies the amount of vertical padding to place around the label of a menu entry. The label and bitmaps are always centered vertically within the menu. Values for this resource are expressed as a percentage of the font's height. The default value (25) increases the default height to 125% of the font's height.

SmeLine

The `SmeLine` object is used to add a horizontal line or menu separator to a `SimpleMenu`. Since each menu entry is an independent object, the application is able to change the color, height, and other attributes of the menu entries, on an entry-by-entry basis. This entry is not selectable, and does not highlight when the pointer cursor is over it.

The class hierarchy for `SmeLine` is: `Object` → `RectObj` → `Sme` → `SmeLine`

Resources

The following new resources are associated with the `SmeLine` object:

`foreground` (class `Foreground`)

The foreground color of the menu entry's window.

`lineWidth` (class `LineWidth`)

The width of the horizontal line to be displayed.

`stipple` (class `Stipple`)

If a bitmap is specified for this resource, the line will be stippled through it. This allows the menu separator to be rendered as something more exciting than just a line. For instance, if the application defines a stipple that is a chain link, then menu separators will look like chains.

StripChart

The `StripChart` widget is used to provide a real-time graphic chart of a single value. This widget is used by `xload` to provide the load graph. It will read data from an application and update the chart at the interval specified by `update`.

The class hierarchy for `StripChart` is: `Core` → `Simple` → `StripChart`

Resources

The following new resources are associated with the StripChart widget:

`highlight` (class `Foreground`)

The color that will be used to draw the scale lines on the graph.

`jumpScroll` (class `JumpScroll`)

When the graph reaches the right edge of the window it must be scrolled to the left. This resource specifies the number of pixels it will jump. Smooth scrolling can be achieved by setting this resource to 1. The default is half the width of the widget.

`minScale` (class `Scale`)

The minimum scale for the graph. The number of divisions on the graph will always be greater than or equal to this value. Default is 1.

`update` (class `Interval`)

The number of seconds between graph updates. Each update is represented on the graph as a 1-pixel-wide line. Every `update` seconds, a new graph point will be added to the right end of the StripChart. Default is 10.

Text

A Text widget is a window that provides a way for an application to display one or more lines of text. The displayed text can reside in a file on disk or in a string in memory. An option also lets an application display a vertical Scrollbar in the Text window, letting you scroll through the displayed text. Other options allow an application to let you modify the text in the window or search for a specific string.

Three types of edit mode are available:

- Append-only
- Read-only
- Editable

Append-only mode lets you enter text into the window, while read-only mode does not. Text may be entered only if the insertion point is after the last character in the window. Editable mode lets you place the cursor anywhere in the text and modify the text at that position. The text cursor position can be modified by using the keystrokes or pointer buttons defined by the event bindings.

The Text widget is designed to separate the storage of text (source) from the painting of the text (sink). The Text widget properly coordinates the sources and sinks. The AsciiText widget is a subclass of the Text widget that automatically creates the source and sink for a client. Most applications will use AsciiText widgets for displaying and editing text.

The class hierarchy for Text is: Core → Simple → Text → AsciiText

Resources

The following new resources are associated with the Text widget:

`autoFill` (class `AutoFill`)

A Boolean that specifies whether the Text widget will automatically break a line when you attempt to type into the right margin. Default is `False`.

`bottomMargin` (class `Margin`)

`topMargin` (class `Margin`)

Amount of space, in pixels, between the edge of the window and the corresponding edge of the text within the window. Default is 2.

`displayCaret` (class `Output`)

A Boolean that specifies whether to display the text caret. Default is `True`.

`displayPosition` (class `TextPosition`)

Character position of the first line. Default is 0.

`insertPosition` (class `TextPosition`)

Character position of the caret. Default is 0.

`leftMargin` (class `Margin`)

`rightMargin` (class `Margin`)

Amount of space, in pixels, between the edge of the window and the corresponding edge of the text within the window. Default is 2.

`resize` (class `Resize`)

Whether the widget should attempt to resize to its preferred dimensions whenever its resources are modified with `XtSetValues()`.

`scrollHorizontal` (class `Scroll`)

`scrollVertical` (class `Scroll`)

Control the placement of scrollbars on the left and bottom edges of the text widget. Possible values are `textScrollAlways`, `textScrollWhenNeeded`, and `textScrollNever` (the default).

`selectTypes` (class `SelectTypes`)

An array of entries that specifies what is highlighted on each successive click in a sequence of multiclicks. Possible values in the array are: `selectAll`, `selectChar`, `selectLine`, `selectNull`, `selectParagraph`, `selectPosition`, and `selectWord`.

The following new resources are associated with the `AsciiText` widget:

`echo` (class `Boolean`)

Whether or not to echo characters to the screen. Default is `True`. This resource is typically set by the application.

`editType` (class `EditType`)

The edit mode of the widget. Possible values are `textAppend`, `textEdit`, and `textRead`. This resource is typically set by the application.

font (class Font)

The font used for the text.

string (class String)

The string for the text source. This resource is typically set by the application.

wrap (class Wrap)

Specifies how text wraps in the widget. Possible values are `textWrapNever`, `textWrapLine`, and `textWrapWord`.

Toggle

The Toggle widget is an area, often rectangular, containing a text or pixmap label. This widget maintains a Boolean state (e.g., True/False or On/Off) and changes state whenever it is selected. When the pointer is on the button, the button border is highlighted to indicate that the button is ready for selection. When pointer button 1 is pressed and released, the Toggle widget indicates that it has changed state by reversing its foreground and background colors, and its `notify` action is invoked. If the pointer is moved out of the widget before the button is released, the widget reverts to its normal foreground and background colors, and releasing the button has no effect. This behavior allows you to cancel an action.

Toggle buttons may also be part of a radio group. A radio group is a list of Toggle buttons in which no more than one Toggle may be set at any time.

The class hierarchy for Toggle is: Core → Simple → Label → Command → Toggle

Resources

The following new resources are associated with the Toggle widget:

radioGroup (class widget)

Specifies another Toggle widget that is in the radio group to which this Toggle widget should be added. A radio group is a group of Toggle widgets, only one of which may be set at a time. If this value is `NULL` (the default), then the Toggle is not part of any radio group and can change state without affecting any other Toggle widgets. If the widget specified in this resource is not already in a radio group, then a new radio group is created containing these two Toggle widgets. No Toggle widget can be in multiple radio groups.

state (class State)

Specifies whether the Toggle widget is set (`True`) or unset (`False`). The default is `False`.

Tree

The Tree widget provides geometry management of arbitrary widgets arranged in a directed, acyclic graph (i.e., a tree). The hierarchy is constructed by attaching a constraint resource called `treeParent` to each child indicating which other node in the tree should be treated as the child's superior. The structure of the tree is shown by laying out the nodes in the standard format for tree diagrams with lines drawn to connect each node with its children.

The Tree sizes itself according to the needs of its children and is not intended to be resized by its parent. Instead, it is typically placed inside another composite widget (such as the Port-hole or Viewport) that can be used to scroll around in the tree.

The class hierarchy for Tree is: Core → Composite → Constraint → Tree

Resources

The following new resources are associated with the Tree widget:

`autoReconfigure` (class `AutoReconfigure`)

Whether or not to lay out the tree every time a node is added or removed. Default is `False`.

`foreground` (class `Foreground`)

Foreground color for the widget.

`gravity` (class `Gravity`)

Specifies the side of the widget from which the tree should grow. Valid values include `WestGravity`, `NorthGravity`, `EastGravity`, and `SouthGravity`.

`hSpace` (class `HSpace`)

`vSpace` (class `VSpace`)

Amount of horizontal and vertical space, in pixels, to leave between the children. This resource also specifies the amount of space between the outermost children and the edge of the box.

`lineWidth` (class `LineWidth`)

The width of the lines drawn between nodes that do not have a `treeGC` constraint resource and their inferiors in the tree.

Tree is a subclass of `Constraint`, which means it has special kind of resources called constraint resources. These resources apply to—and are specified as if they belong to—the children of the Tree rather than to the Tree itself. Tree specifies the following constraint resource for its children:

`treeParent` (class `TreeParent`)

This specifies the superior node in the tree for this widget. The default is for the node to have no superior (and to therefore be at the top of the tree).

Viewport

The Viewport widget consists of a frame window, one or two Scrollbars, and an inner window (usually containing a child widget). The size of the frame window is determined by the viewing size of the data that is to be displayed and the dimensions to which the Viewport is created. The inner window is the full size of the data that is to be displayed and is clipped by the frame window. The Viewport widget controls the scrolling of the data directly.

When the geometry of the frame window is equal in size to the inner window, or when the data does not require scrolling, the Viewport widget automatically removes any scroll bars. The `forceBars` resource causes the Viewport widget to display any scroll bar permanently.

The class hierarchy for Viewport is: Core → Composite → Constraint → Viewport

Resources

The following new resources are associated with the Viewport widget:

`allowHoriz` (class Boolean)

`allowVert` (class Boolean)

Flags to allow horizontal and vertical scroll bars. Default values are `False`. Setting the resource to `True` allows a Viewport child to increase in size horizontally or vertically.

`forceBars` (class Boolean)

Flag to force display of scroll bars. Default value is `False`. Normally, when the geometry of the frame window is equal in size to the inner window, or when the data does not require scrolling, Viewport automatically removes any scroll bars. Setting `forceBars` to `True` causes the Viewport widget to display any scroll bar permanently.

`useBottom` (class Boolean)

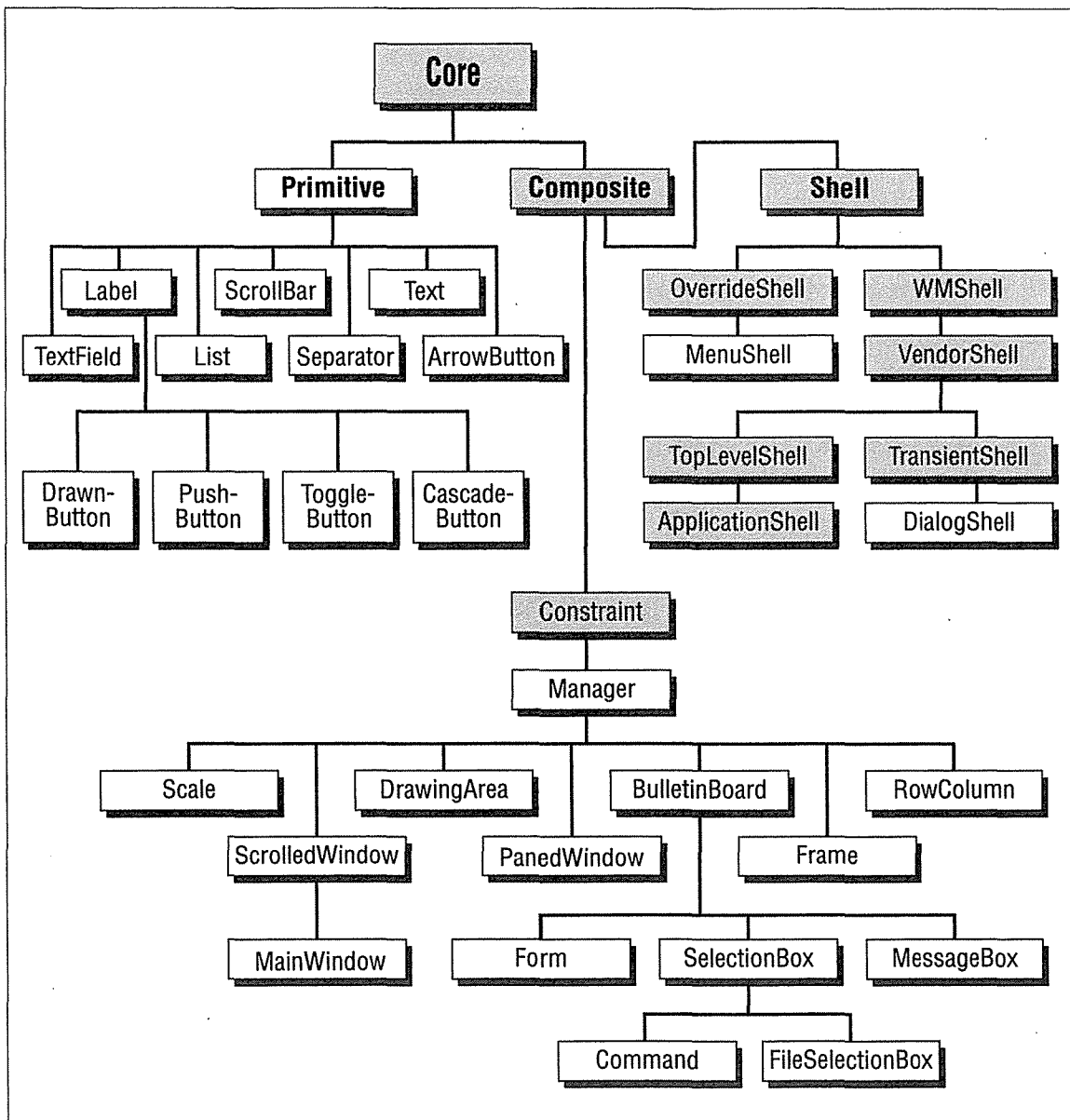
Flag to indicate whether the horizontal scrollbar is placed at the bottom or the top of the widget. Default is `False`, meaning to put the scrollbar on top.

`useRight` (class Boolean)

Flag to indicate whether the vertical scrollbar is placed at the right or the left of the widget. Default is `False`, meaning to put the scrollbar on the left.

Motif Widget Resources

Commercial Motif applications build their user interfaces with components from the Motif widget set. Figure G-4 shows the complete class hierarchy of the Motif widgets. The widgets shown in gray are defined by the X Toolkit intrinsics, and are common to all Xt-based widget sets.



Widget Resources

Figure G-4. Inheritance among the Motif widgets

Figure G-4 does not include the gadgets that are part of the Motif class hierarchy: LabelGadget, SeparatorGadget, ArrowButtonGadget, CascadeButtonGadget, PushButtonGadget, and ToggleButtonGadget. As mentioned before, gadgets are essentially windowless widgets.

All of the Motif gadgets are subclassed from the Motif Gadget class, which is subclassed from RectObj.

The rest of this section provides more detailed information on each of the Motif widgets and gadgets and their resources. For each widget, there is a brief description and a list of the new resources defined by the widget. Since gadgets define the same new resources as their corresponding widgets, we've included the information on each gadget in the material for the relevant widget. Note that these resource lists include only those resources that can be set in resource files; they are not complete lists. For the full reference material on the Motif widgets, see Volume Six, *Motif Programming Manual*.

ArrowButton

An ArrowButton is a directional arrow-shaped button that includes a shaded border. The shading changes to make the ArrowButton appear either pressed in when selected or raised when unselected.

The class hierarchy for ArrowButton is: Core → XmPrimitive → XmArrowButton

ArrowButtonGadget is the gadget variant of ArrowButton. It has the same appearance and behavior as an ArrowButton. The class hierarchy for ArrowButtonsGadget is: Object → RectObj → XmGadget → XmArrowButtonGadget

Resources

The following new resource is associated with both the ArrowButton and ArrowButtonGadget:

arrowDirection (class ArrowDirection)
Sets the arrow direction. Possible values are ARROW_UP, ARROW_LEFT, ARROW_DOWN, ARROW_RIGHT.

BulletinBoard

BulletinBoard is a general-purpose manager that allows children to be placed at arbitrary x,y positions. The simple geometry management of BulletinBoard can be used to enforce margins and to prevent child widgets from overlapping. BulletinBoard is the base widget for most dialog widgets and defines many resources that have an effect only when it is an immediate child of a DialogShell.

The class hierarchy for BulletinBoard is: Core → Composite → Constraint → XmManager → XmBulletinBoard

Resources

The following new resources are associated with the `BulletinBoard` widget:

`allowOverlap` (class `AllowOverlap`)

If `True` (default), child widgets are allowed to overlap.

`buttonFontList` (class `ButtonFontList`)

The font list used for the button children of the `BulletinBoard` widget. If this value is initially `NULL`, the font list is derived from the `buttonFontList` resource found in the nearest ancestor that is a subclass of `BulletinBoard`, `VendorShell`, or `MenuShell`.

`defaultPosition` (class `DefaultPosition`)

If `True` (default) and if the `BulletinBoard` is the child of a `DialogShell`, then the `BulletinBoard` is centered relative to the `DialogShell`'s parent.

`dialogTitle` (class `DialogTitle`)

The dialog title. Setting this resource also sets the resources `title` and `titleEncoding` in a parent that is a subclass of `WMShell`.

`labelFontList` (class `LabelFontList`)

Like the `buttonFontList` resource, but for `Label` children.

`marginHeight` (class `MarginHeight`)

`marginWidth` (class `MarginWidth`)

Minimum spacing between a `BulletinBoard`'s top or bottom edge and any child widget or its right or left edge and any child widget.

`noResize` (class `NoResize`)

If `False` (default), `mwm` includes resize controls in the window manager frame of the `BulletinBoard`'s shell parent.

`resizePolicy` (class `ResizePolicy`)

How `BulletinBoard` widgets are resized. Possible values are `RESIZE_NONE`, which means that the widget remains at a fixed size; `RESIZE_GROW`, which means that the widget only expands; and `RESIZE_ANY`, which means that the widget shrinks or expands as needed.

`shadowType` (class `ShadowType`)

The style in which shadows are drawn. Possible values are `SHADOW_IN`, `SHADOW_OUT`, `SHADOW_ETCHED_IN`, and `SHADOW_ETCHED_OUT`.

`textFontList` (class `TextFontList`)

Like the `buttonFontList` resource, but for `Text` children.

CascadeButton

`CascadeButtons` are used in menu systems to post menus. A `CascadeButton` either links a menu bar to a menu pane or connects a menu pane to another menu pane. The widget can have a menu attached to it as a submenu.

The class hierarchy for CascadeButton is: Core → XmPrimitive → XmLabel → XmCascadeButton

CascadeButtonGadget is the gadget variant of CascadeButton. It has the same appearance and behavior as a CascadeButton. The class hierarchy for CascadeButtonsGadget is: Object → RectObj → XmGadget → XmLabelGadget → XmCascadeButtonGadget

Resources

The following new resources are associated with the CascadeButton and CascadeButtonGadget:

`cascadePixmap` (class `Pixmap`)

The pixmap within the CascadeButton that indicates a submenu. By default, this pixmap is an arrow pointing toward the submenu to be popped up.

`mappingDelay` (class `MappingDelay`)

The number of milliseconds it should take for the application to display a submenu after its CascadeButton has been selected.

`subMenuId` (class `MenuWidget`)

The widget ID of the pulldown menu pane associated with the CascadeButton. The menu pane is displayed when the CascadeButton is selected. This resource is typically set by the application.

Command

Command is a composite widget that handles command entry by providing a prompt, a command input field, and a history list region. Many of the Command widget's new resources are in fact renamed resources from SelectionBox.

The class hierarchy for Command is: Core → Composite → Constraint → XmManager → XmBulletinBoard → XmSelectionBox → XmCommand

Resources

The following new resources are associated with the Command widget:

`command` (class `TextString`)

The text currently displayed on the command line.

`historyItems` (class `Items`)

The items in the history list. This value is not typically set in resource files.

`historyItemCount` (class `ItemCount`)

The number of strings in `historyItems`. This value is not typically set in resource files.

historyMaxItems (class MaxItems)

The history list's maximum number of items. When this number is reached, the first history item is removed before the new command is added to the list.

historyVisibleItemCount (class VisibleItemCount)

The number of history list commands that will display at one time.

promptString (class PromptString)

The command-line prompt.

DialogShell

DialogShell is the parent for dialog boxes. A DialogShell cannot be iconified separately, but only when the main application shell is iconified. The child of a DialogShell is typically a subclass of BulletinBoard and much of the functionality of DialogShell is based on this assumption.

The class hierarchy for DialogShell is: Core → Composite → Shell → WmShell → VendorShell → TransientShell → XmDialogShell

Resources

DialogShell does not have any new user-settable resources associated with it.

Display

In Motif 1.2, the Display object stores display-specific information for use by the toolkit. An application has a Display object for each display it accesses. When an application creates its first shell on a display a Display object is created automatically.

The dragInitiatorProtocolStyle and dragReceiverProtocolStyle resources specify the drag protocol for an application that performs drag and drop operations. The two protocol styles are dynamic and preregister. Under the dynamic protocol, the initiator and receiver pass messages back and forth to handle drag and drop visuals. Under the preregister protocol, the initiator handles drag and drop visuals by reading information that is preregistered and stored in properties. The actual protocol that is used by a specific initiator and receiver is based on the requested protocol styles of the receiver and initiator:

Drag Initiator Protocol Style	Drag Receiver Protocol Style			
	Preregister	Prefer Preregister	Prefer Dynamic	Dynamic
Preregister	PREREGISTER	PREREGISTER	PREREGISTER	DROP_ONLY
Prefer Preregister	PREREGISTER	PREREGISTER	PREREGISTER	DYNAMIC
Prefer Receiver	PREREGISTER	PREREGISTER	DYNAMIC	DYNAMIC
Prefer Dynamic	PREREGISTER	DYNAMIC	DYNAMIC	DYNAMIC
Dynamic	DROP_ONLY	DYNAMIC	DYNAMIC	DYNAMIC

The class hierarchy for Display is: Core → Core → Composite → Shell → WmShell → VendorShell → TopLevelShell → ApplicationShell → XmDisplay

Resources

The following new resources are associated with the Display widget:

defaultVirtualBindings (class DefaultVirtualBindings)

The default virtual bindings for the display.

dragInitiatorProtocolStyle (class DragInitiatorProtocolStyle)

The client's drag and drop protocol requirements or preference when it is the initiator of a drag and drop operation. Possible values are DRAG_PREREGISTER, DRAG_DYNAMIC, DRAG_NONE, DRAG_DROP_ONLY, DRAG_PREFER_DYNAMIC, DRAG_PREFER_PREREGISTER, and DRAG_PREFER_RECEIVER.

dragReceiverProtocolStyle (class DragReceiverProtocolStyle)

The client's drag and drop protocol requirements or preference when it is the receiver. Possible values are DRAG_PREREGISTER, DRAG_DYNAMIC, DRAG_NONE, DRAG_DROP_ONLY, DRAG_PREFER_DYNAMIC, and DRAG_PREFER_PREREGISTER.

DrawingArea

DrawingArea provides a blank canvas for interactive drawing. The widget does not do any drawing of its own. Since DrawingArea is a subclass of Manager, it can provide simple geometry management of multiple widget or gadget children.

The class hierarchy for DrawingArea is: Core → Composite → Constraint → XmManager → XmDrawingArea

Resources

The following new resources are associated with the DrawingArea widget:

marginHeight (class MarginHeight)

marginWidth (class MarginWidth)

The spacing between a DrawingArea's top or bottom edge and any child widget or its right or left edge and any child widget.

resizePolicy (class ResizePolicy)

How DrawingArea widgets are resized. Possible values are RESIZE_NONE, which means that the widget remains at a fixed size; RESIZE_GROW, which means that the widget only expands; and RESIZE_ANY, which means that the widget shrinks or expands as needed.

DrawnButton

DrawnButton is an empty widget window, surrounded by a shaded border. The widget provides a graphics area that can act like a PushButton. The graphics can be dynamically updated by the application.

The class hierarchy for DrawnButton is: Core → XmPrimitive → XmLabel → XmDrawn-Button

Resources

The following new resources are associated with the DrawnButton widget:

pushButtonEnabled (class PushButtonEnabled)

If False (default), the shadow drawing doesn't appear three dimensional; if True, the shading provides a pushed in or raised appearance as for the PushButton widget.

shadowType (class ShadowType)

The style in which shadows are drawn. Possible values are SHADOW_IN, SHADOW_OUT, SHADOW_ETCHED_IN, and SHADOW_ETCHED_OUT.

FileSelectionBox

FileSelectionBox is a composite widget that is used to traverse a directory hierarchy and select files. FileSelectionBox provides a directory mask input field, a scrollable list of sub-directories, a scrollable list of filenames, a filename input field, and a group of four PushButtons. The PushButtons are typically labeled OK, Filter, Cancel, and Help by default.

The class hierarchy for FileSelectionBox is: Core → Composite → Constraint → Xm-Manager → XmBulletinBoard → XmSelectionBox → XmFileSelectionBox

Resources

The following new resources are associated with the FileSelectionBox widget:

directory (class Directory)

The base directory that, in combination with pattern, forms the directory mask (the dirMask resource). The directory mask determines which files and directories to display.

disListLabelString (class DirListLabelString)

The string that labels the directory list.

dirMask (class DirMask)

The directory mask that determines which files and directories to display. This value combines the values of the resources directory and pattern.

`dirSpec` (class `DirSpec`)

The complete specification of the file path. It is the initial directory and file search that determines the default value for this resource.

`fileListLabelString` (class `fileListLabelString`)

The string that labels the file list.

`fileTypeMask` (class `FileTypeMask`)

Determines whether the file list will display only regular files, only directories, or any type of file. Possible values are `FILE_DIRECTORY`, `FILE_REGULAR`, and `FILE_ANY_TYPE`.

`filterLabelString` (class `FilterLabelString`)

The string that labels the field in which the directory mask is typed in by the user.

`noMatchString` (class `NoMatchString`)

A string that displays in the file list when there are no filenames to display.

`pattern` (class `Pattern`)

The file search pattern that, in combination with `directory`, forms the directory mask (the `dirMask` resource). The directory mask determines which files and directories to display. If the `pattern` resource defaults to `NULL` or is empty, a pattern for matching all files will be used.

Form

Form is a container widget that constrains its children so as to define their layout when the Form is resized. Constraints on the children of a Form specify the attachments for each of the four sides of a child. Children may be attached to each other, to edges of the Form, or to relative positions within the Form.

The class hierarchy for Form is: Core → Core → Composite → Constraint → XmManager → XmBulletinBoard → XmForm

Resources

The following new resources are associated with the Form widget:

`fractionBase` (class `MaxValue`)

The denominator part of the fraction that describes a child's relative position within a Form. The numerator of this fraction is one of the four positional constraint resources: `bottomPosition`, `leftPosition`, `rightPosition`, or `topPosition`. This resource is typically set by the application.

`horizontalSpacing` (class `Spacing`)

The offset for right and left attachments.

`rubberPositioning` (class `RubberPositioning`)

Defines the default behavior of a child's top and left side, in the absence of other settings. If this resource is `False` (default), the child's top and left sides are positioned

using absolute values. If True, the child's top and left sides are positioned relative to the size of the Form.

`verticalSpacing` (class `VerticalSpacing`)

The offset for top and bottom attachments.

Form is a subclass of `Constraint`, which means it has special kind of resources called constraint resources. These resources apply to—and are specified as if they belong to—the children of the Form rather than to the Form itself. Form specifies the following constraint resources for its children:

`bottomAttachment` (class `Attachment`)

`leftAttachment` (class `Attachment`)

`rightAttachment` (class `Attachment`)

`topAttachment` (class `Attachment`)

The method of attachment for each of the child's sides. Possible values are `ATTACH_NONE` (side remains unattached), `ATTACH_FORM` (side is attached to same edge of the Form), `ATTACH_OPPOSITE_FORM` (side is attached to other edge of the Form), `ATTACH_WIDGET` (side abuts an adjacent widget), `ATTACH_OPPOSITE_WIDGET` (side is attached to other edge of an adjacent widget), `ATTACH_POSITION` (side is placed relative to a dimension of the Form), and `ATTACH_SELF` (side is placed relative to its current position and to the Form). Each attachment refers to the corresponding edge of the child widget within the Form. These resource are typically set by an application.

`bottomOffset` (class `Offset`)

`leftOffset` (class `Offset`)

`rightOffset` (class `Offset`)

`topOffset` (class `Offset`)

The distance between the child's side and the object it's attached to. Offsets are absolute. A nonzero offset is ignored when an attachment resource is set to `ATTACH_POSITION` because a resize operation applies relative positioning in this case.

`bottomPosition` (class `Attachment`)

`leftPosition` (class `Attachment`)

`rightPosition` (class `Attachment`)

`topPosition` (class `Attachment`)

Used in conjunction with `fractionBase` to calculate the position of the side of a child, relative to that side of the Form. These resources have no effect unless the child's particular attachment resource is set to `ATTACH_POSITION`. These resources are typically set by the application.

`bottomWidget` (class `Widget`)

`leftWidget` (class `Widget`)

`rightWidget` (class `Widget`)

`topWidget` (class `Widget`)

The name of the widget or gadget that serves as the attachment point for the particular side of the child. To use these resources, set the particular attachment resource to either `ATTACH_WIDGET` or `ATTACH_OPPOSITE_WIDGET`. These resources are typically set by the application.

`resizable` (class `Boolean`)

If `True` (default), a child's resize request is accepted by the `Form`, provided that the child isn't constrained by its attachments. That is, if both the left and right sides of a child are attached, or if both the top and bottom are attached, the resize request fails, whereas if the child has only one horizontal or one vertical attachment, the resize request is granted. If this resource is `False`, the child is never resized.

Frame

`Frame` is a simple subclass of `Manager` that places a three-dimensional border around a single child. `Frame` is used to provide the typical Motif-style appearance for widget classes that do not have a visible frame, such as `RowColumn`. As of Motif 1.2, a `Frame` can have two children: a work area child and a title child. The widget uses constraint resources to indicate the type of each child and to specify the alignment of the title child.

The class hierarchy for `Frame` is: `Core` → `Composite` → `Constraint` → `XmManager` → `XmFrame`

Resources

The following new resources are associated with the `Frame` widget:

`marginHeight` (class `MarginHeight`)

`marginWidth` (class `MarginWidth`)

The spacing between a `DrawingArea`'s top or bottom edge and any child widget or its right or left edge and any child widget.

`resizePolicy` (class `ResizePolicy`)

How `DrawingArea` widgets are resized. Possible values are `RESIZE_NONE`, which means that the widget remains at a fixed size; `RESIZE_GROW`, which means that the widget only expands; and `RESIZE_ANY`, which means that the widget shrinks or expands as needed.

`Frame` is a subclass of `Constraint`, which means it has special kind of resources called constraint resources. These resources apply to—and are specified as if they belong to—the children of the `Frame` rather than to the `Frame` itself. `Frame` specifies the following constraint resources for its children:

`childType` (class `ChildType`)

The type of the child. `Frame` supports one title and one work area child. Possible values are `FRAME_TITLE_CHILD`, `FRAME_WORKAREA_CHILD`, and `FRAME_GENERIC_CHILD`. This value is typically set by the application.

`childHorizontalAlignment` (class `ChildHorizontalAlignment`)

The alignment (left to right) for a `Frame`'s title. Possible values are `ALIGNMENT_BEGINNING`, `ALIGNMENT_CENTER`, and `ALIGNMENT_END`.

`childHorizontalSpacing` (class `childHorizontalSpacing`)

The minimum distance between the title text and the Frame shadow. The title is clipped to maintain this distance. The value of `marginWidth` is used as the default value.

`childVerticalPlacement` (class `ChildVerticalPlacement`)

The alignment of the Frame's title relative to the top shadow of the Frame. Possible values are `ALIGNMENT_BASELINE_BOTTOM`, `ALIGNMENT_BASELINE_TOP`, `ALIGNMENT_WIDGET_TOP`, `ALIGNMENT_CENTER`, and `ALIGNMENT_WIDGET_BOTTOM`.

Gadget

Gadget is a supporting superclass for other gadget classes. A Gadget never appears in an application, but it does define resources that are inherited by its subclasses. Gadget takes care of drawing and highlighting border shadows as well as managing traversal. A gadget uses its Manager widget parent's pixmap and color resources (e.g., foreground). If you change such a resource in a manager widget, all of its gadget children will be affected as well.

The class hierarchy for Gadget is: `Object` → `RectObj` → `XmGadget`

Resources

The following new resources are associated with Gadget:

`highlightOnEnter` (class `HighlightOnEnter`)

Determines whether to draw a gadget's highlighting rectangle whenever the cursor moves into the gadget. This resource applies only when the shell has a focus policy of `POINTER`. If the `highlightOnEnter` resource is `True`, highlighting is drawn; if `False` (default), highlighting is not drawn.

`highlightThickness` (class `HighlightThickness`)

The thickness of the highlighting rectangle.

`navigationType` (class `NavigationType`)

Determines the way in which gadgets are to be traversed during keyboard navigation. Possible values are `NONE`, `TAB_GROUP`, `STICKY_TAB_GROUP`, and `EXCLUSIVE_TAB_GROUP`. This value is typically set by the application.

`shadowThickness` (class `ShadowThickness`)

The thickness of the shadow border.

`traversalOn` (class `TraversalOn`)

If `True` (default), traversal of this gadget is made possible.

`unitType` (class `UnitType`)

The measurement units to use in resources that specify a size or position—for example, any resources of data type `Dimension` (whose names generally include one of the words “Margin” or “Thickness”). For a gadget whose parent is a Manager subclass, the default value is copied from this parent (provided the value hasn't been

explicitly set by the application); otherwise, the default is `PIXELS`. Possible values are `PIXELS`, `100TH_POINTS`, `100TH_MILLIMETERS`, `100TH_FONT_UNITS`, and `1000TH_INCHES`. This value is typically set by the application.

Label

Label provides a text string or a pixmap for labeling other widgets in an application. Label is also a superclass for the various button widgets. Label does not accept any button or key events, but it does receive enter and leave events.

The class hierarchy for Label is: `Core` → `XmPrimitive` → `XmLabel`

LabelGadget is the gadget variant of Label. It has the same appearance and behavior as a Label. The class hierarchy for LabelGadget is: `Object` → `RectObj` → `XmGadget` → `XmLabelGadget`

Resources

The following new resources are associated with the Label and LabelGadget:

`accelerator` (class `Accelerator`)

A string that describes a button widget's accelerator (the modifiers and key to use as a shortcut in selecting the button). The string's format is like that of a translation but allows only a single key press event to be specified.

`acceleratorText` (class `AcceleratorText`)

The text that is displayed for an accelerator. This value is typically set by an application.

`alignment` (class `Alignment`)

The alignment (left to right) for a label's text or pixmap. Possible values are `ALIGNMENT_BEGINNING`, `ALIGNMENT_CENTER`, and `ALIGNMENT_END`.

`fontList` (class `FontList`)

The font list used for the widget's text. If this value is initially `NULL`, the font list is derived from the `labelFontList` or `buttonFontList` resource from the nearest parent that is a subclass of `BulletinBoard`, `MenuShell`, or `VendorShell`.

`labelInsensitivePixmap` (class `LabelInsensitivePixmap`)

The pixmap label for an insensitive button (when `labelType` is `PIXMAP`).

`labelPixmap` (class `LabelPixmap`)

The pixmap used when `labelType` is `PIXMAP`.

`labelString` (class `labelString`)

The string used for the label when `labelType` is `STRING`. If this resource is `NULL`, the application uses the widget's name.

`labelType` (class `labelType`)

The type of label (either string or pixmap). Possible values are `PIXMAP` and `STRING`. This value is typically set by an application.

`marginBottom` (class `MarginBottom`)

`marginLeft` (class `MarginLeft`)

`marginRight` (class `MarginRight`)

`marginTop` (class `MarginTop`)

The amount of space between one side of the label text and the nearest margin.

`marginHeight` (class `MarginHeight`)

`marginWidth` (class `MarginWidth`)

The spacing between one side of the label and the nearest edge of a shadow.

`mnemonic` (class `Mnemonic`)

A keysym that gives you another way to select a button. In the label string, the first character matching this keysym will be underlined.

`mnemonicCharSet` (class `mnemonicCharSet`)

The character set for the label's mnemonic.

`recomputeSize` (class `RecomputeSize`)

If `True` (default), the `Label` widget changes its size so that the string or pixmap fits exactly.

`stringDirection` (class `StringDirection`)

The direction in which to draw the string. Possible values are `STRING_DIRECTION_L_TO_R` and `STRING_DIRECTION_R_TO_L`.

List

`List` provides a list of choices from which you can select one or more items, based on the selection policy. `List` supports four selection policies: `Single Select`, `Browse Select`, `Multiple Select`, and `Extended Select`.

In `Single Select` mode, only one item can be selected at a time; a button press on an item selects it and deselects the previously selected item. In `Browse Select` mode, only one item can be selected at a time; a button press works as in `Single Select` mode and, additionally, a button drag moves the selection with the pointer. In `Multiple Select` mode, any number of items can be selected at a time; a button press toggles the selection state of an item and does not change the selection state of any other items. In `Extended Select` mode, any number of items can be selected at a time; discontinuous ranges of items can be selected by combining button presses and button drags.

The class hierarchy for `List` is: `Core` → `XmPrimitive` → `XmList`

Resources

The following new resources are associated with the List widget:

`automaticSelection` (class `AutomaticSelection`)

If `True` (and the widget's `selectionPolicy` is either `BROWSE_SELECT` or `EXTENDED_SELECT`), then the selection takes effect whenever you move into a new item. If `False`, then you must release the mouse button before any selection takes effect.

`doubleClickInterval` (class `DoubleClickInterval`)

The time span (in milliseconds) within which two button clicks must occur to be considered a double click rather than two single clicks. By default, this value is the multi-click time of the display.

`fontList` (class `FontList`)

The font list used for the items in the list. If this value is initially `NULL`, the font list is derived from the `textFontList` resource from the nearest parent that is a subclass of `BulletinBoard` or `VendorShell`.

`itemCount` (class `ItemCount`)

The total number of items. This value is not normally set in a resource file.

`items` (class `Items`)

The list items to display.

`listMarginHeight` (class `ListMarginHeight`)

`listMarginWidth` (class `ListMarginWidth`)

The height or width of the margin between the border of the list and the items in the list.

`listSizePolicy` (class `listSizePolicy`)

The method for resizing the widget when a list item exceeds the width of the work area. Possible values are `VARIABLE`, `CONSTANT`, and `RESIZE_IF_POSSIBLE`.

`listSpacing` (class `ListSpacing`)

The spacing between items.

`scrollBarDisplayPolicy` (class `ScrollBarDisplayPolicy`)

Determines when to display vertical scrollbars in a `ScrolledList` widget. Possible values are `STATIC` and `AS_NEEDED`.

`selectedItemCount` (class `SelectedItemCount`)

The number of items in the list of selected items. This value is not normally set in a resource file.

`selectedItems` (class `SelectedItems`)

The currently selected list items. This value is not normally set in a resource file.

`selectionPolicy` (class `SelectionPolicy`)

Determines the effect of a selection action. Possible values are `SINGLE_SELECT`, `BROWSE_SELECT`, `MULTIPLE_SELECT`, and `EXTENDED_SELECT`. This value is typically set by the application.

`stringDirection` (class `StringDirection`)

The direction in which to draw the string. Possible values are `STRING_DIRECTION_L_TO_R` and `STRING_DIRECTION_R_TO_L`.

`topItemPosition` (class `TopItemPosition`)

The position of the first item that will be visible in the list. The first position is specified as 1 and the last position is specified as 0.

`visibleItemCount` (class `VisibleItemCount`)

The number of items to display in the work area of the list.

MainWindow

`MainWindow` provides the standard appearance for the primary window of an application. `MainWindow` supports five standard areas: a menu bar, a command window, a work region, a message window, and two scrollbars (one horizontal and one vertical). An application can use as many or as few of these areas as necessary; they are all optional. A `MainWindow` can also create up to three `Separator` widgets for dividing one area from another.

The class hierarchy for `MainWindow` is: `Core` → `Composite` → `Constraint` → `XmManager` → `XmScrolledWindow` → `XmMainWindow`

Resources

The following new resources are associated with the `MainWindow` widget:

`commandWindowLocation` (class `CommandWindowLocation`)

One of two positions for the command window. Possible values are `COMMAND_ABOVE_WORKSPACE` and `COMMAND_BELOW_WORKSPACE`.

`mainWindowMarginHeight` (class `mainWindowMarginHeight`)

`mainWindowMarginWidth` (class `mainWindowMarginWidth`)

The margin on the top or bottom (right or left) of the `MainWindow` widget. These resources override the corresponding margin resources in the `ScrolledWindow` widget.

Manager

`Manager` is a superclass for Motif widget classes that contain children. The `Manager` widget never appears in applications, but it does define resources that are inherited by its subclasses. `Manager` supports geometry management by providing resources for visual shadows and highlights and for keyboard traversal mechanisms.

The default values of the color resources for the foreground, background, top and bottom shadows, and highlighting are set dynamically. If no colors are specified, they are generated automatically. On a monochrome system, black and white colors are selected. On a color system, four colors are selected that provide the appropriate shading for the 3-D visuals.

When the background color is specified, the shadow colors are selected to provide the appropriate 3-D appearance and foreground, and highlight colors are selected to provide the necessary contrast.

The class hierarchy for Manager is: Core → Composite → Constraint → XmManager

Resources

The following new resources are associated with the Manager widget:

`bottomShadowColor` (class `BottomShadowColor`)

The color used in drawing the border shadow's bottom and right sides on a color display.

`bottomShadowPixmap` (class `bottomShadowPixmap`)

The pixmap used in drawing the border shadow's bottom and right sides on a monochrome display.

`foreground` (class `Foreground`)

The foreground color used by Manager widgets.

`highlightColor` (class `HighlightColor`)

The color used in drawing the highlighting rectangle on a color display.

`highlightPixmap` (class `HighlightPixmap`)

The pixmap used in drawing the highlighting rectangle on a monochrome display.

`navigationType` (class `NavigationType`)

Determines the way in which gadgets are to be traversed during keyboard navigation. Possible values are `NONE`, `TAB_GROUP`, `STICKY_TAB_GROUP`, and `EXCLUSIVE_TAB_GROUP`. This value is typically set by the application.

`shadowThickness` (class `ShadowThickness`)

The thickness of the shadow border.

`stringDirection` (class `StringDirection`)

The direction in which to draw the string. Possible values are `STRING_DIRECTION_L_TO_R` and `STRING_DIRECTION_R_TO_L`.

`topShadowColor` (class `TopShadowColor`)

The color used in drawing the border shadow's top and left sides on a color display.

`topShadowPixmap` (class `TopShadowPixmap`)

The pixmap used in drawing the border shadow's top and left sides on a monochrome display.

`traversalOn` (class `TraversalOn`)

If `True` (default), traversal of this widget is made possible.

`unitType` (class `UnitType`)

The measurement units to use in resources that specify a size or position—for example, any resources of data type `Dimension` (whose names generally include one of the words “Margin” or “Thickness”). For a gadget whose parent is a Manager subclass, the default value is copied from this parent (provided the value hasn't been

explicitly set by the application); otherwise, the default is `PIXELS`. Possible values are `PIXELS`, `100TH_POINTS`, `100TH_MILLIMETERS`, `100TH_FONT_UNITS`, and `1000TH_INCHES`. This value is typically set by the application.

MenuShell

`MenuShell` is a subclass of `OverrideShell` that is meant to contain only popup or pulldown menu panes.

The class hierarchy for `MenuShell` is: `Core` → `Composite` → `Shell` → `OverrideShell` → `XmMenuShell`

Resources

The following new resources are associated with the `MenuShell` widget:

`buttonFontList` (class `ButtonFontList`)

In Motif 1.2, the font list used for the button children of the `MenuShell` widget. If this value is initially `NULL` and if the value of `defaultFontList` is not `NULL`, this value is used. Otherwise, the font list is derived from the `buttonFontList` resource found in the nearest ancestor that is a subclass of `BulletinBoard`, `VendorShell`, or `MenuShell`.

`defaultFontList` (class `DefaultFontList`)

The default font list for the children of the `MenuShell` widget. This resource is obsolete in Motif 1.2.

`labelFontList` (class `LabelFontList`)

Like the `buttonFontList` resource, but for `Label` children.

MessageBox

`MessageBox` is composite widget that is used for creating simple message dialog boxes, which normally present transient messages. A `MessageBox` usually contains a message symbol, a message, three `PushButtons`, and a separator between the message and the buttons. The `dialogType` resource controls the type of message symbol that is displayed. The `PushButtons` are typically labeled `OK`, `Cancel`, and `Help` by default.

The class hierarchy for `MessageBox` is: `Core` → `Composite` → `Constraint` → `XmManager` → `XmBulletinBoard` → `XmMessageBox`

Resources

The following new resources are associated with the `MessageBox` widget:

`cancelLabelString` (class `CancelLabelString`)

The string that labels the Cancel button.

`defaultButtonType` (class `defaultButtonType`)

Specifies which `PushButton` provides the default action. Possible values are `DIALOG_CANCEL_BUTTON`, `DIALOG_OK_BUTTON`, and `DIALOG_HELP_BUTTON`.

`dialogType` (class `DialogType`)

The type of `MessageBox` dialog, which also indicates the message symbol that displays by default. Possible values are `DIALOG_ERROR`, `DIALOG_INFORMATION`, `DIALOG_MESSAGE`, `DIALOG_QUESTION`, `DIALOG_TEMPLATE`, `DIALOG_WARNING`, and `DIALOG_WORKING`. This value is typically set by an application.

`helpLabelString` (class `HelpLabelString`)

The string that labels the Help button.

`messageAlignment` (class `MessageAlignment`)

The type of alignment for the message label. Possible values are `ALIGNMENT_BEGINNING`, `ALIGNMENT_CENTER`, and `ALIGNMENT_END`.

`messageString` (class `MessageString`)

The string to use as the message label.

`minimizeButtons` (class `MinimizeButtons`)

If `False` (default), all buttons are standardized to be as wide as the widest button and as high as the highest button. If `True`, buttons will keep their preferred size.

`okLabelString` (class `OkLabelString`)

The string that labels the OK button.

`symbolPixmap` (class `SymbolPixmap`)

The pixmap label to use as the message symbol.

PanedWindow

`PanedWindow` is a constraint widget that tiles its children vertically. A `PanedWindow` is as wide as its widest child and all children are made that width. Users can adjust the height of a pane using a sash that appears below the corresponding pane.

The class hierarchy for `PanedWindow` is: `Core` → `Composite` → `Constraint` → `XmManager` → `XmPanedWindow`

Resources

The following new resources are associated with the `PanedWindow` widget:

`marginHeight` (class `MarginHeight`)

The spacing between a `PanedWindow` widget's top or bottom edge and any child widget.

`marginWidth` (class `MarginWidth`)

The spacing between a `PanedWindow` widget's right or left edge and any child widget.

`refigureMode` (class `RefigureMode`)

If `True` (default), children are reset to their appropriate positions following a change in the `PanedWindow` widget.

`sashHeight` (class `SashHeight`)

`sashWidth` (class `SashWidth`)

The height and width of the sash.

`sashIndent` (class `SashIndent`)

The horizontal position of the sash along each pane. Positive values specify the indent from the left edge; negative values, from the right edge (assuming the default value of `stringDirection`). If the value is too large, the sash is placed flush with the edge of the `PanedWindow`.

`sashShadowThickness` (class `SashShadowThickness`)

The thickness of shadows drawn on each sash.

`separatorOn` (class `SeparatorOn`)

If `True`, the widget places a `Separator` or `SeparatorGadget` between each pane.

`spacing` (class `Spacing`)

The distance between each child pane.

`PanedWindow` is a subclass of `Constraint`, which means it has special kind of resources called constraint resources. These resources apply to—and are specified as if they belong to—the children of the `PanedWindow` rather than to the `PanedWindow` itself. `PanedWindow` specifies the following constraint resource for its children:

`allowResize` (class `AllowResize`)

If `False` (default), the `PanedWindow` widget always refuses resize requests from its children. If `True`, the `PanedWindow` widget tries to grant requests to change a child's height.

`paneMaximum` (class `PaneMaximum`)

`paneMinimum` (class `PaneMinimum`)

The values of a pane's maximum and minimum dimensions for resizing. You can prevent a sash from being drawn by setting these values to be equal.

`positionIndex` (class `PositionIndex`)

In Motif 1.2, the position of the widget in the `PanedWindow`'s list of children, not including sashes. A value of 0 indicates the beginning of the list, while `LAST_POSITION` places the child at the end of the list.

skipAdjust (class SkipAdjust)

If `False` (default), the `PanedWindow` widget automatically resizes this pane child. If `True`, resizing is not automatic, and the `PanedWindow` may choose to skip the adjustment of this pane.

Primitive

`Primitive` is a supporting superclass that provides Motif-specific resources for border drawing, highlighting, and keyboard traversal mechanisms. The `Primitive` widget never appears in applications, but it does define resources that are inherited by its subclasses. `Primitive` supports widget subclasses that handle elementary graphic elements such as buttons, labels, and separators.

The default values of the color resources for the foreground, background, top and bottom shadows, and highlighting are set dynamically. If no colors are specified, they are generated automatically. On a monochrome system, black and white colors are selected. On a color system, four colors are selected that provide the appropriate shading for the 3-D visuals. When the background color is specified, the shadow colors are selected to provide the appropriate 3-D appearance and foreground and highlight colors are selected to provide the necessary contrast.

The class hierarchy for `Primitive` is: `Core` → `XmPrimitive`

Resources

The following new resources are associated with the `Primitive` widget:

`bottomShadowColor` (class `BottomShadowColor`)

The color used in drawing the border shadow's bottom and right sides on a color display.

`bottomShadowPixmap` (class `bottomShadowPixmap`)

The pixmap used in drawing the border shadow's bottom and right sides on a monochrome display.

`foreground` (class `Foreground`)

The foreground color used by `Primitive` widgets.

`highlightColor` (class `HighlightColor`)

The color used in drawing the highlighting rectangle on a color display.

`highlightOnEnter` (class `HighlightOnEnter`)

Determines whether to draw the widget's highlighting rectangle whenever the cursor moves into the widget. This resource applies only when the shell has a focus policy of `POINTER`. If the `highlightOnEnter` resource is `True`, highlighting is drawn; if `False` (default), highlighting is not drawn.

`highlightPixmap` (class `HighlightPixmap`)

The pixmap used in drawing the highlighting rectangle on a monochrome display.

`highlightThickness` (class `HighlightThickness`)

The thickness of the highlighting rectangle.

`navigationType` (class `NavigationType`)

Determines the way in which a Primitive widget is traversed during keyboard navigation. Possible values are `NONE`, `TAB_GROUP`, `STICKY_TAB_GROUP`, and `EXCLUSIVE_TAB_GROUP`. This value is typically set by the application.

`shadowThickness` (class `ShadowThickness`)

The thickness of the shadow border.

`topShadowColor` (class `TopShadowColor`)

The color used in drawing the border shadow's top and left sides on a color display.

`topShadowPixmap` (class `TopShadowPixmap`)

The pixmap used in drawing the border shadow's top and left sides on a monochrome display.

`traversalOn` (class `TraversalOn`)

If `True` (default), traversal of this widget is made possible.

`unitType` (class `UnitType`)

The measurement units to use in resources that specify a size or position—for example, any resources of data type `Dimension` (whose names generally include one of the words “Margin” or “Thickness”). For a widget whose parent is a `Manager` subclass, the default value is copied from this parent (provided the value hasn't been explicitly set by the application); otherwise, the default is `PIXELS`. Possible values are `PIXELS`, `100TH_POINTS`, `100TH_MILLIMETERS`, `100TH_FONT_UNITS`, and `1000TH_INCHES`. This value is typically set by the application.

PushButton

A `PushButton` is a widget that causes something to happen in an application. A `PushButton` displays a textual or graphics label. It invokes an application callback when it is clicked on with the mouse. The shading of the `PushButton` changes to make it appear either pressed in when selected or raised when unselected.

The class hierarchy for `PushButton` is: `Core` → `XmPrimitive` → `XmLabel` → `XmPushButton`

`PushButtonGadget` is the gadget variant of `PushButton`. It has the same appearance and behavior as a `PushButton`. The class hierarchy for `PushButtonsGadget` is: `Object` → `RectObj` → `XmGadget` → `XmLabelGadget` → `XmPushButtonGadget`

Resources

The following new resources are associated with `PushButton` and `PushButtonGadget`:

`armColor` (class `ArmColor`)

The color with which the armed button is filled. For a color display, the default color is a shade between the bottom shadow color and the background color. For a monochrome display, the default is the foreground color, and label text is switched to the background color. This resource is in effect only when `fillOnArm` is set to `True`.

`armPixmap` (class `ArmPixmap`)

The pixmap that identifies the button when it is armed (and when its `labelType` is `PIXMAP`). For a `PushButton` in a menu, this resource is disabled.

`defaultButtonShadowThickness` (class `DefaultButtonShadowThickness`)

The width of the shadow used to indicate a default `PushButton`.

`fillOnArm` (class `FillOnArm`)

If `True` (default), the `PushButton` widget fills the button (when armed) with the color specified by `armColor`. If `False`, the `PushButton` widget only switches the top and bottom shadow colors. For a `PushButton` in a menu, this resource is disabled (and assumed to be `False`).

`showAsDefault` (class `ShowAsDefault`)

Indicates the default `PushButton` by displaying a shadow. (In a menu, this resource is disabled.)

RowColumn

`RowColumn` provides an area in which children belonging to any widget type are displayed in rows and columns. `RowColumn` is a general-purpose manager widget class that can be configured into many layouts, such as a `MenuBar`, `PopupMenu`, `PulldownMenu`, `OptionMenu`, `CheckBox`, or `RadioBox`. Many of `RowColumn`'s resources pertain only to a specific layout type.

In Motif 1.2, a `RowColumn` that is configured as a `PopupMenu` or a `PulldownMenu` supports tear-off menus. When a menu is torn off, it remains on the screen after a selection is made so that additional selections can be made. A menu pane that can be torn off contains a tear-off button at the top of the menu that contains a dashed line.

The class hierarchy for `RowColumn` is: `Core` → `Composite` → `Constraint` → `XmManager` → `XmRowColumn`

Resources

The following new resources are associated with the RowColumn widget:

`adjustLast` (class `AdjustLast`)

If `True` (default), the last row (or column) in the RowColumn widget is expanded so as to be flush with the edge.

`adjustMargin` (class `AdjustMargin`)

If `True` (default), text in each row (or column) will align with other text in its row (or column). This is done by forcing the margin resources (defined by the Label widget) to have the same value.

`entryAlignment` (class `EntryAlignment`)

When `isAligned` is `True`, this resource tells RowColumn children how to align. The children must be subclasses of `XmLabel` or `XmLabelGadget`. Possible values are `ALIGNMENT_BEGINNING`, `ALIGNMENT_CENTER`, and `ALIGNMENT_END`.

`entryBorder` (class `EntryBorder`)

The border width of a RowColumn widget's children.

`entryClass` (class `EntryClass`)

The widget class to which children must belong when being added to a RowColumn widget. This resource is used only when the `isHomogeneous` resource is set to `True`. This value is not normally set in a resource file.

`entryVerticalAlignment` (class `EntryVerticalAlignment`)

In Motif 1.2, specifies how children that are subclasses of `Label`, `Text`, and `TextField` are aligned vertically. The resource has no effect if `orientation` is `VERTICAL` or `packing` is `PACK_TIGHT`. Possible values are `ALIGNMENT_BASELINE_BOTTOM`, `ALIGNMENT_BASELINE_TOP`, `ALIGNMENT_CONTENTS_BOTTOM`, `ALIGNMENT_CENTER`, and `ALIGNMENT_CONTENTS_TOP`.

`isAligned` (class `IsAligned`)

If `True`, enables the alignment specified in the `entryAlignment` resource. Alignment is ignored in a label whose parent is a popup or pulldown `MenuPane`.

`isHomogeneous` (class `IsHomogeneous`)

If `True`, enforces the condition that all RowColumn children belong to the same class (the class specified by the `entryClass` resource).

`labelString` (class `LabelString`)

A label used only in option menus. A text string displays beside the selection area. By default, there is no label.

`marginHeight` (class `MarginHeight`)

`marginWidth` (class `MarginWidth`)

The spacing between an edge of the RowColumn widget and its nearest child. In popup and pulldown menus, the default is 0; in other types of RowColumn widgets, the default is 3 pixels.

`menuAccelerator` (class `MenuAccelerator`)

A pointer to a string that specifies an accelerator (keyboard shortcut) for use only in RowColumn widgets of type `MENU_POPUP` or `MENU_BAR`. In a popup menu, typing the accelerator posts the menu; in a menu bar, typing the accelerator highlights the first item and enables traversal in the menu bar.

`menuPost` (class `MenuPost`)

The string that describes the event for posting a menu. The default value depends on the type of RowColumn widget: for `MENU_POPUP`, the default is `BMenu Press`; for `MENU_OPTION`, `MENU_BAR`, and `WORK_AREA` the default is `BSelect Press`; for `MENU_PULLDOWN`, this resource isn't meaningful.

`mnemonic` (class `Mnemonic`)

The keysym of the key to press (in combination with the `MAIt` modifier) in order to post the pulldown menu associated with an option menu. This resource is meaningful only in option menus. In the label string, the first character matching this keysym will be underlined.

`mnemonicCharSet` (class `MnemonicCharSet`)

The character set for the option menu's mnemonic.

`numColumns` (class `NumColumns`)

The number of columns (in a vertically-oriented RowColumn widget) or the number of rows (in a horizontally-oriented RowColumn widget). This resource is meaningful only when the `packing` resource is set to `PACK_COLUMN`.

`orientation` (class `Orientation`)

The direction for laying out the rows and columns of children of a RowColumn widget. Possible values are `VERTICAL` and `HORIZONTAL`. This value is typically set by the application.

`packing` (class `Packing`)

The method of spacing the items placed within a RowColumn widget. Possible values are `PACK_TIGHT`, which means that each child widget is set to its minimum size; `PACK_COLUMN`, which means that the size of each child widget is padded so that they are all the same size; and `PACK_NONE`, which means that the RowColumn accommodates the size and location of each child widget. This value is typically set by the application.

`popupEnabled` (class `PopupEnabled`)

If `True` (default), keyboard shortcuts are in effect for popup menus. Set this resource to `False` if you want to disable accelerators and mnemonics in popup menus.

`radioAlwaysOne` (class `RadioAlwaysOne`)

This resource is effective only when the `radioBehavior` resource is `True`. `radioAlwaysOne`, when set to `True` (default), ensures that one of the toggle buttons is always selected.

`radioBehavior` (class `radioBehavior`)

If `True`, the RowColumn widget acts like a `RadioBox` by setting two of the resources for its toggle button children. Namely, the `indicatorType` resource defaults to

ONE_OF_MANY, and the `visibleWhenOff` resource defaults to `True`. This value is typically set by the application.

`resizeHeight` (class `ResizeHeight`)

`resizeWidth` (class `ResizeWidth`)

If `True` (default), the widget requests a new height or width when necessary. If `False`, no resize requests are made.

`rowColumnType` (class `RowColumnType`)

The type of `RowColumn` widget to create. Possible values are `WORK_AREA`, `MENU_PULLDOWN`, `MENU_BAR`, `MENU_OPTION`, and `MENU_POPUP`. This value is typically set by the application.

`spacing` (class `Spacing`)

The horizontal and vertical spacing between children in the `RowColumn` widget.

`subMenuId` (class `SubMenuId`)

The widget ID for the pulldown menu pane to be associated with an `OptionMenu`. This resource is meaningful only in `RowColumn` widgets of type `MENU_OPTION`. This resource is typically set by the application.

`tearOffModel` (class `TearOffModel`)

In Motif 1.2, specifies whether tear-off behavior is enabled for a `RowColumn` with `rowColumnType` set to `MENU_PULLDOWN` or `MENU_POPUP`. Possible values are `TEAR_OFF_DISABLED` and `TEAR_OFF_ENABLED`.

`RowColumn` is a subclass of `Constraint`, which means it has special kinds of resources called constraint resources. These resources apply to—and are specified as if they belong to—the children of the `RowColumn` rather than to the `RowColumn` itself. `RowColumn` specifies the following constraint resource for its children:

`positionIndex` (class `PositionIndex`)

In Motif 1.2, the position of the widget in the `RowColumn`'s list of children. A value of 0 indicates the beginning of the list, while `LAST_POSITION` places the child at the end of the list.

Scale

A `Scale` displays a value from a range of values and allows a user to adjust the value. A `Scale` consists of a narrow, rectangular trough that contains a slider. The slider's position marks the current value within the range of values. `Scale` is a manager widget that orients its children along its axis. These children, typically labels, can be used as tick marks. If the `Scale` widget is an input-output type (`sensitive is True`), you can change the value by moving the slider. An output-only `Scale` displays a value but does not allow you to modify it.

The class hierarchy for `Scale` is: `Core` → `Composite` → `Constraint` → `XmManager` → `XmScale`

Resources

The following new resources are associated with the Scale widget:

`decimalPoints` (class `DecimalPoints`)

A positive integer that determines how the slider's value will be displayed. The decimal point in the slider's value gets shifted to the right, and this resource specifies the number of decimal places to shift. For example, if the slider's value is 1234, then setting the `decimalPoints` resource to 2 causes the widget to display the value as 12.34.

`fontList` (class `FontList`)

The font list used for the text specified by the `titleString` resource. If this value is initially `NULL`, the font list is derived from the font list resource from the nearest parent that is a subclass of `BulletinBoard`, `MenuShell`, or `VendorShell`.

`highlightOnEnter` (class `HighlightOnEnter`)

Determines whether to draw the widget's highlighting rectangle whenever the cursor moves into the widget. This resource applies only when the shell has a focus policy of `POINTER`. If the `highlightOnEnter` resource is `True`, highlighting is drawn; if `False` (default), highlighting is not drawn.

`highlightThickness` (class `HighlightThickness`)

The thickness of the highlighting rectangle.

`maximum` (class `Maximum`)

`minimum` (class `Minimum`)

The maximum/minimum value of the slider.

`orientation` (class `Orientation`)

The direction in which the scale is displayed. Possible values are `VERTICAL` and `HORIZONTAL`.

`processingDirection` (class `ProcessingDirection`)

Determines the position at which to display the slider's maximum and minimum values, with respect to the slider. Possible values are `MAX_ON_TOP`, `MAX_ON_BOTTOM`, `MAX_ON_LEFT`, and `MAX_ON_RIGHT`.

`scaleHeight` (class `ScaleHeight`)

`scaleWidth` (class `ScaleWidth`)

The height or width of the slider area.

`scaleMultiple` (class `ScaleMultiple`)

The distance to move the slider when the user moves it by a multiple increment. The default value is calculated as $(\text{maximum} - \text{minimum}) / 10$.

`showValue` (class `ShowValue`)

If `True`, the label specifying the slider's current value will be displayed beside the slider. If `False`, the label isn't displayed.

`titleString` (class `TitleString`)

The text string that appears as the title in the Scale widget.

value (class Value)

The current position of the slider along the scale. This resource must have a value between the values of minimum and maximum. This value is not normally set in a resource file.

Screen

In Motif 1.2, the Screen object stores screen-specific information for use by the toolkit. An application has a Screen object for each screen that it accesses. When an application creates its first shell on a screen, a Screen object is created automatically.

The class hierarchy for Screen is: Core → XmScreen

Resources

The following new resources are associated with the Screen widget:

darkThreshold (class DarkThreshold)

The level of perceived brightness (between 0 and 100) that is treated as a “dark” background color when computing default shadow and select colors.

defaultCopyCursorIcon (class DefaultCopyCursorIcon)

The DragIcon used during a copy operation. When the value is NULL, a default system icon is used.

defaultInvalidCursorIcon (class DefaultInvalidCursorIcon)

The DragIcon used when the pointer is over an invalid drop site. When the value is NULL, a default system icon is used.

defaultLinkCursorIcon (class DefaultLinkCursorIcon)

The DragIcon used during a link operation. When the value is NULL, a default system icon is used.

defaultMoveCursorIcon (class DefaultMoveCursorIcon)

The DragIcon used during a move operation. When the value is NULL, a default system icon is used.

defaultNoneCursorIcon (class DefaultNoneCursorIcon)

The DragIcon used when the pointer is not over a drop site. When the value is NULL, a default system icon is used.

defaultSourceCursorIcon (class DefaultSourceCursorIcon)

The bitmap used as a cursor when a sourceCursorIcon is not provided by the DragContext. When the value is NULL, a default system icon is used.

defaultValidCursorIcon (class DefaultValidCursorIcon)

The DragIcon used when the pointer is over a valid drop site. When the value is NULL, a default system icon is used.

`font` (class `Font`)

The font used in computing values for `horizontalFontUnit` and `verticalFontUnit`.

`foregroundThreshold` (class `ForegroundThreshold`)

The level of perceived brightness (between 0 and 100) that distinguishes between a “dark” and “light” background when computing the default foreground and highlight colors.

`horizontalFontUnits` (class `HorizontalFontUnits`)

The horizontal component of the font units that are used to convert geometry values when `shellUnitType` or `unitType` is set to `100TH_FONT_UNITS`. If a value is not specified, the default is computed from the `font` resource.

`lightThreshold` (class `LightThreshold`)

The level of perceived brightness (between 0 and 100) that is treated as a “light” background color when computing default shadow and select colors.

`menuCursor` (class `MenuCursor`)

The cursor that is used when the application posts a menu. Possible values include all of the cursor in the X cursor font.

`moveOpaque` (class `MoveOpaque`)

If `False` (default), an operation that moves a window displays an outline of the window. If `True`, a move operation displays a representation of the window.

`verticalFontUnits` (class `VerticalFontUnits`)

The vertical component of the font units that are used to convert geometry values when `shellUnitType` or `unitType` is set to `100TH_FONT_UNITS`. If a value is not specified, the default is computed from the `font` resource.

ScrollBar

A `ScrollBar` allows you to reposition data that is too large to fit in the viewing window. Although a `ScrollBar` can be used as a standalone widget, it is normally used in a `ScrolledWindow`. A `ScrollBar` consists of a rectangular strip, called the scroll region or trough, and two arrows placed on either end of the scroll region. Within the scroll region is a smaller, movable rectangle called the slider. To scroll the data, you can click on one of the arrows, click in the scroll region, or drag the slider. The application typically sets the `sliderSize` resource such that the size of the slider relative to the size of the scroll region corresponds to the percentage of total data that is currently displayed.

The class hierarchy for `ScrollBar` is: `Core` → `Core` → `XmPrimitive` → `XmScrollBar`

Resources

The following new resources are associated with the ScrollBar widget:

`increment` (class `Increment`)

The amount the value changes due to the user's moving the slider one increment.

`initialDelay` (class `InitialDelay`)

The number of milliseconds a button must remain pressed before triggering continuous slider movement.

`maximum` (class `Maximum`)

`minimum` (class `Minimum`)

The maximum/minimum value of the slider.

`orientation` (class `Orientation`)

The direction in which the scale is displayed. Possible values are `VERTICAL` and `HORIZONTAL`.

`pageIncrement` (class `PageIncrement`)

The amount the value changes due to the user's moving the slider one page increment.

`processingDirection` (class `ProcessingDirection`)

Determines the position at which to display the slider's maximum and minimum values, with respect to the slider. Possible values are `MAX_ON_TOP`, `MAX_ON_BOTTOM`, `MAX_ON_LEFT`, and `MAX_ON_RIGHT`.

`repeatDelay` (class `RepeatDelay`)

The number of milliseconds a button must remain pressed before continuing further slider motions, once the `initialDelay` time has been triggered.

`showArrows` (class `ShowArrows`)

If `True`, arrows are displayed; if `False`, they are not.

`sliderSize` (class `SliderSize`)

The slider's length. The length ranges from 1 to the value of `maximum - minimum`.

`troughColor` (class `TroughColor`)

The color of the slider's trough.

`value` (class `Value`)

The slider's position. The position ranges from the value of `minimum` to the value of `(maximum - sliderSize)`. This value is not normally set in a resource file.

ScrolledWindow

`ScrolledWindow` provides a scrollable view of data that may not be visible all at once. `ScrollBars` allow you to scroll the visible part of the window through the larger display. A `ScrolledWindow` widget can be created so that it scrolls automatically without application intervention or so that an application provides support for all scrolling operations.

The class hierarchy for ScrolledWindow is: Core → Composite → Constraint → XmManager → XmScrolledWindow

Resources

The following new resources are associated with the ScrolledWindow widget:

`scrollBarDisplayPolicy` (class `ScrollBarDisplayPolicy`)

Controls the placement of ScrollBars, depending on the value of the `scrollingPolicy` resource. Possible values are `STATIC` and `AS_NEEDED`. If `scrollingPolicy` is set to `AUTOMATIC`, then `scrollBarDisplayPolicy` defaults to a value of `AS_NEEDED`, and ScrollBars are displayed only when the workspace cannot fit within the clip area. If `scrollingPolicy` is set to `APPLICATION_DEFINED`, then `scrollBarDisplayPolicy` defaults to (and must remain with) a value of `STATIC`. This means that ScrollBars will always be displayed.

`scrollBarPlacement` (class `ScrollBarPlacement`)

The positions of the ScrollBars relative to the work window. The default value of this resource depends on the value of the `stringDirection` resource. Possible values are `TOP_LEFT`, `BOTTOM_LEFT`, `TOP_RIGHT`, and `BOTTOM_RIGHT`.

`scrolledWindowMarginHeight` (class `scrolledWindowMarginHeight`)

The spacing at the top and bottom of the ScrolledWindow.

`scrolledWindowMarginWidth` (class `scrolledWindowMarginWidth`)

The spacing at the right and left sides of the ScrolledWindow.

`scrollingPolicy` (class `ScrollingPolicy`)

Determines how automatic scrolling occurs. Possible values are `AUTOMATIC` and `APPLICATION_DEFINED`. This value is typically set by the application.

`spacing` (class `Spacing`)

The distance between each ScrollBar and the work window.

SelectionBox

`SelectionBox` is a composite widget that displays a scrollable list of alternatives from which you can choose items. A `SelectionBox` contains a text field in which you can enter a selection, the scrollable list of selections, labels for the text field and the scrollable list, a separator, and a group of three or four buttons. The `PushButtons` are typically labeled `OK`, `Apply`, `Cancel`, and `Help` by default.

The class hierarchy for `SelectionBox` is: Core → Composite → Constraint → XmManager → XmBulletinBoard → XmSelectionBox

Resources

The following new resources are associated with the SelectionBox widget:

`applyLabelString` (class `ApplyLabelString`)

The string that labels the Apply button.

`cancelLabelString` (class `CancelLabelString`)

The string that labels the Cancel button.

`childPlacement` (class `ChildPlacement`)

In Motif 1.2, determines the placement of the work area child. Possible values are `PLACE_ABOVE_SELECTION`, `PLACE_BELOW_SELECTION`, and `PLACE_TOP`. This value is typically set by the application.

`dialogType` (class `DialogType`)

Determines the type of SelectionBox widget that will be initially created and managed. Possible values are `DIALOG_WORK_AREA`, `DIALOG_PROMPT`, `DIALOG_SELECTION`, `DIALOG_COMMAND`, and `DIALOG_FILE_SELECTION`. This value is typically set by the application.

`helpLabelString` (class `HelpLabelString`)

The string that labels the Help button.

`listItemCount` (class `ItemCount`)

The number of items in the SelectionBox list. This value is not normally set in a resource file.

`listItems` (class `Items`)

The items in the SelectionBox list.

`listLabelString` (class `ListLabelString`)

The string that labels the SelectionBox list.

`listVisibleItemCount` (class `VisibleItemCount`)

The number of items that appear in the SelectionBox list.

`minimizeButtons` (class `MinimizeButtons`)

If `False` (default), all buttons are standardized to be as wide as the widest button and as high as the highest button. If `True`, buttons will keep their preferred size.

`mustMatch` (class `MustMatch`)

If `True`, the selection that a user types in the text edit field must match an existing entry in the SelectionBox list. If `False` (default), the typed selection doesn't need to match a list entry. This value is typically set by the application.

`okLabelString` (class `OkLabelString`)

The string that labels the Ok button.

`selectionLabelString` (class `SelectionLabelString`)

The string that labels the text edit field.

`textAccelerators` (class `TextAccelerators`)

The translations to add to the SelectionBox's Text widget child. The default bindings allow the up and down keys to be used in selecting list items.

`textColumns` (class `Columns`)

The number of columns in the Text widget.

`textString` (class `TextString`)

The text string that appears in the text edit selection field.

Separator

A Separator is a widget that draws a horizontal or vertical line between components in an application. Several line styles are available for the Separator. A pixmap separator can also be made by specifying a pixmap for the Core resource `backgroundPixmap` and then setting `separatorType` to `NO_LINE`.

The class hierarchy for Separator is: `Core` → `XmPrimitive` → `XmSeparator`

`SeparatorGadget` is the gadget variant of Separator. It has the same appearance and behavior as a Separator. The class hierarchy for `SeparatorGadget` is: `Object` → `RectObj` → `XmGadget` → `XmSeparatorGadget`

Resources

The following new resources are associated with Separator and SeparatorGadget:

`margin` (class `Margin`)

The spacing on either end of the Separator.

`orientation` (class `Orientation`)

The direction in which to display the Separator. Possible values are `VERTICAL` and `HORIZONTAL`.

`separatorType` (class `SeparatorType`)

The line style in which to draw the Separator. Possible values are `NO_LINE`, `SINGLE_LINE`, `DOUBLE_LINE`, `SINGLE_DASHED_LINE`, `DOUBLE_DASHED_LINE`, `SHADOW_ETCHED_IN`, and `SHADOW_ETCHED_OUT`.

Text

The Text widget provides a text editor that allows text to be inserted, modified, deleted, and selected. Text provide both single-line and multiline text editing capabilities.

The class hierarchy for Text is: `Core` → `XmPrimitive` → `XmText`

Resources

The following new resources are associated with the Text widget:

`autoShowCursorPosition` (class `AutoShowCursorPosition`)

If `True` (default), the visible portion of the Text widget will always contain the insert cursor. The Text widget will scroll its contents, if necessary, to ensure that the cursor remains visible.

`blinkRate` (class `BlinkRate`)

The time in milliseconds that the cursor spends either being visible or invisible. A value of 0 prevents the cursor from blinking.

`columns` (class `Columns`)

The number of character spaces that should fit horizontally in the text window. The width resource determines the default value of `columns`, but if no width has been set, the default is 20.

`cursorPosition` (class `CursorPosition`)

The location at which to place the current insert cursor. Values for this resource are relative to the beginning of the text; the first character position is defined as 0.

`cursorPositionVisible` (class `CursorPositionVisible`)

If `True` (default), the text cursor will be visible.

`editable` (class `Editable`)

If `True` (default), the user is allowed to edit the text string; if `False`, the user is not allowed to do so. This value is typically set by the application.

`editMode` (class `EditMode`)

Determines whether the Text widget is single-line or multiline. Possible values are `MULTI_LINE_EDIT` and `SINGLE_LINE_EDIT`.

`fontList` (class `FontList`)

The font list used for the widget's text. If this value is initially `NULL`, the font list is derived from the font list resource from the nearest parent that is a subclass of `BulletinBoard`, `MenuShell`, or `VendorShell`.

`marginHeight` (class `MarginHeight`)

`marginWidth` (class `MarginWidth`)

The spacing between the edges of the widget and the text.

`maxLength` (class `MaxLength`)

The maximum length of the text string that a user can enter from the keyboard.

`pendingDelete` (class `PendingDelete`)

If `True` (default), the Text widget's pending delete mode is on, meaning that selected text will be deleted as soon as the next text insertion occurs.

`resizeHeight` (class `ResizeHeight`)

If `False` (default), the Text widget will not expand vertically to fit all of the text (in other words, the widget will need to have scrollbars so that the rest of the text can be scrolled into view). If `True`, the Text widget always begins its display with the text at

the beginning of the source. This resource has no effect in a Text widget inside of a ScrolledWindow whose scrollVertical resource is set to True.

`resizeWidth` (class `ResizeWidth`)

If `False` (default), the Text widget will not expand horizontally to fit its text. If `True`, the widget tries to change its width. This resource has no effect when the `wordWrap` resource is set to `True`.

`rows` (class `Rows`)

The number of character spaces that should fit vertically in the text window. The `height` resource determines the default value of `rows`, but if no `height` has been set, the default is 1. This resource is meaningful only when `editMode` is `MULTI_LINE_EDIT`.

`scrollHorizontal` (class `Scroll`)

When a Text widget is inside a ScrolledWindow, if `True`, the Text widget adds a horizontal ScrollBar. This resource is meaningful only when `editMode` is `MULTI_LINE_EDIT`.

`scrollLeftSide` (class `ScrollSide`)

When a Text widget is inside a ScrolledWindow, if `True`, the vertical ScrollBar is placed to the left of the scrolled text window. This resource is meaningful only when `editMode` is `MULTI_LINE_EDIT` and when `scrollVertical` is `True`.

`scrollTopSide` (class `ScrollSide`)

When a Text widget is inside a ScrolledWindow, if `True`, the horizontal ScrollBar is placed above the scrolled text window, rather than below by default.

`scrollVertical` (class `Scroll`)

When a Text widget is inside a ScrolledWindow, if `True`, the Text widget adds a vertical ScrollBar.

`selectionArray` (class `SelectionArray`)

The array of possible actions caused by multiple mouse clicks. Possible values are `SELECT_POSITION`, `SELECT_WORD`, `SELECT_LINE`, and `SELECT_ALL`.

`selectionArrayCount` (class `SelectionArrayCount`)

The number of items in the array specified by `selectionArray`.

`selectionThreshold` (class `SelectionThreshold`)

The number of pixels the insertion cursor must be dragged during selection in order to select the next character.

`topCharacter` (class `TopCharacter`)

The location of the text to display at the top of the window. Values for this resource are relative to the beginning of the text, with the first character position defined as 0.

`value` (class `Value`)

The string value to display in the Text widget, expressed as a `char *`. If `value` and `valueWcs` are both defined, `valueWcs` takes precedence.

`valueWcs` (class `ValueWcs`)

In Motif 1.2, the string value to display in the `Text` widget, expressed as a `wchar_t *`. If `value` and `valueWcs` are both defined, `valueWcs` takes precedence. This value cannot be set in a resource file.

`verifyBell` (class `VerifyBell`)

If `True` (default), a bell will sound when a verification produces no action.

`wordWrap` (class `WordWrap`)

If `False` (default), do not break lines automatically between words (in which case text can disappear beyond the window's edge). If `True`, break lines at spaces, tabs, or newlines. This resource is meaningful only when `editMode` is `MULTI_LINE_EDIT`.

TextField

The `TextField` widget provides a single-line text editor that has a subset of the functionality of the `Text` widget.

The class hierarchy for `TextField` is: `Core` → `XmPrimitive` → `XmTextField`

Resources

The following new resources are associated with the `TextField` widget:

`blinkRate` (class `BlinkRate`)

The time in milliseconds that the cursor spends either being visible or invisible. A value of 0 prevents the cursor from blinking.

`columns` (class `Columns`)

The number of character spaces that should fit horizontally in the text window. The `width` resource determines the default value of `columns`, but if no `width` has been set, the default is 20.

`cursorPosition` (class `CursorPosition`)

The location at which to place the current insert cursor. Values for this resource are relative to the beginning of the text; the first character position is defined as 0.

`cursorPositionVisible` (class `CursorPositionVisible`)

If `True` (default), the text cursor will be visible.

`editable` (class `Editable`)

If `True` (default), the user is allowed to edit the text string; if `False`, the user is not allowed to do so. This value is typically set by the application.

`fontList` (class `FontList`)

The font list used for the widget's text. If this value is initially `NULL`, the font list is derived from the font list resource from the nearest parent that is a subclass of `BulletinBoard`, `MenuShell`, or `VendorShell`.

marginHeight (class MarginHeight)

marginWidth (class MarginWidth)

The spacing between the edges of the widget and the text.

maxLength (class MaxLength)

The maximum length of the text string that a user can enter from the keyboard.

pendingDelete (class PendingDelete)

If True (default), the TextField widget's pending delete mode is on, meaning that selected text will be deleted as soon as the next text insertion occurs.

resizeWidth (class ResizeWidth)

If False (default), the TextField widget will not expand horizontally to fit its text. If True, the widget tries to change its width.

selectionArray (class SelectionArray)

The array of possible actions caused by multiple mouse clicks. Possible values are SELECT_POSITION, SELECT_WORD, and SELECT_LINE.

selectionArrayCount (class SelectionArrayCount)

The number of items in the array specified by selectionArray.

selectionThreshold (class SelectionThreshold)

The number of pixels the insertion cursor must be dragged during selection in order to select the next character.

value (class Value)

The string value to display in the TextField widget, expressed as a char *. If value and valueWcs are both defined, valueWcs takes precedence.

valueWcs (class ValueWcs)

In Motif 1.2, the string value to display in the TextField widget, expressed as a wchar_t *. If value and valueWcs are both defined, valueWcs takes precedence. This value cannot be set in a resource file.

verifyBell (class VerifyBell)

If True (default), a bell will sound when a verification produces no action.

ToggleButton

A ToggleButton is a button that is either set or unset. ToggleButtons are typically used in groups, called RadioBoxes and CheckBoxes, depending on the behavior of the buttons. In a RadioBox, a ToggleButton displays *one-of-many* behavior, which means that only one button in the group can be set at a time. When a button is selected, the previously selected button is unset. In a CheckBox, a ToggleButton displays *n-of-many* behavior, which means that any number of ToggleButtons can be set at one time. ToggleButton uses an indicator to show its state; the shape of the indicator specifies the type of behavior. A diamond-shaped indicator is used for one-of-many ToggleButtons and a square-shaped indicator is used for *n-of-many* ToggleButtons.

The class hierarchy for `ToggleButton` is: `Core` → `XmPrimitive` → `XmLabel` → `XmToggleButton`

`ToggleButtonGadget` is the gadget variant of `ToggleButton`. It has the same appearance and behavior as a `ToggleButton`. The class hierarchy for `ToggleButtonsGadget` is: `Object` → `RectObj` → `XmGadget` → `XmLabelGadget` → `XmToggleButtonGadget`

Resources

The following new resources are associated with `ToggleButton` and `ToggleButtonGadget`:

`fillOnSelect` (class `FillOnSelect`)

If `True`, selection of this `ToggleButton` fills the indicator with the color given by the `selectColor` resource and switches the button's top and bottom shadow colors. If `False`, only the top and bottom shadow colors are switched.

`indicatorOn` (class `IndicatorOn`)

If `True` (default), the indicator is visible and its shadows are switched when the button is toggled. If `False`, the indicator is invisible and no space is set aside for it; in addition, the shadows surrounding the button are switched when it is toggled.

`indicatorSize` (class `IndicatorSize`)

The size of the indicator. This value changes if the size of the button's text string or pixmap changes.

`indicatorType` (class `IndicatorType`)

Determines whether the indicator is drawn as a diamond (one-of-many) or a square (n-of-many). Possible values are `N_OF_MANY` and `ONE_OF_MANY`. This value is typically set by the application.

`selectColor` (class `SelectColor`)

The color with which to fill the indicator when the button is selected.

`selectInsensitivePixmap` (class `SelectInsensitivePixmap`)

The pixmap used for an insensitive `ToggleButton` when it's selected. This resource is meaningful only when the `Label` resource `labelType` is set to `PIXMAP`.

`selectPixmap` (class `SelectPixmap`)

The pixmap used for a `ToggleButton` when it's selected. An unselected `ToggleButton` uses the pixmap specified by the `Label` resource `labelPixmap`. This resource is meaningful only when the `Label` resource `labelType` is set to `PIXMAP`.

`set` (class `Set`)

The selection state of the button.

`spacing` (class `Spacing`)

The distance between the toggle indicator and its label.

`visibleWhenOff` (class `VisibleWhenOff`)

If `True`, the toggle indicator remains visible when the button is unselected.

VendorShell

VendorShell is a vendor-specific supporting superclass for all shell classes that are visible to the window manager and that do not have override redirection. VendorShell defines resources that provide the Motif look and feel and manages the specific communication needed by the Motif window manager (mwm).

The class hierarchy for VendorShell is: Core → Composite → Shell → WmShell → VendorShell

Resources

The following new resources are associated with the VendorShell widget:

`audibleWarning` (class `AudibleWarning`)

In Motif 1.2, specifies whether an action performs an associated audible cue. Possible values are `BELL` and `NONE`.

`buttonFontList` (class `ButtonFontList`)

In Motif 1.2, the font list used for the button children of the VendorShell widget. If this value is initially `NULL` and if the value of `defaultFontList` is not `NULL`, this value is used. Otherwise, the font list is derived from the `buttonFontList` resource found in the nearest ancestor that is a subclass of `BulletinBoard`, `VendorShell`, or `MenuShell`.

`defaultFontList` (class `DefaultFontList`)

The default font list for the children of the MenuShell widget. This resource is obsolete in Motif 1.2.

`deleteResponse` (class `DeleteResponse`)

The action to perform when the shell receives a `WM_DELETE_WINDOW` message. Possible values are `DESTROY`, `UNMAP`, and `DO_NOTHING`. This value is not normally set in a resource file.

`inputMethod` (class `InputMethod`)

In Motif 1.2, specifies the string that sets the locale modifier for the input method.

`keyboardFocusPolicy` (class `KeyboardFocusPolicy`)

The method of assigning keyboard focus. Possible values are `EXPLICIT` and `POINTER`.

`labelFontList` (class `LabelFontList`)

In Motif 1.2, like the `buttonFontList` resource, but for Label children.

`mwmDecorations` (class `MwmDecorations`)

This resource corresponds to the values assigned by the `decorations` field of the `_MOTIF_WM_HINTS` property. This resource determines which frame buttons and handles to include with a window. Possible values are `MWM_DECOR_ALL`, `MWM_DECOR_BORDER`, `MWM_DECOR_RESIZEH`, `MWM_DECOR_TITLE`, `MWM_DECOR_SYSTEM`, `MWM_DECOR_MINIMIZE`, and `MWM_DECOR_MAXIMIZE`.

`mwmFunctions` (class `MwmFunctions`)

This resource corresponds to the values assigned by the `functions` field of the `_MOTIF_WM_HINTS` property. This resource determines which functions to include in the system menu. Possible values are `MWM_FUNC_ALL`, `MWM_FUNC_RESIZE`, `MWM_FUNC_MOVE`, `MWM_FUNC_MINIMIZE`, `MWM_FUNC_MAXIMIZE`, and `MWM_FUNC_CLOSE`.

`mwmInputMode` (class `MwmInputMode`)

This resource corresponds to the values assigned by the `input_mode` field of the `_MOTIF_WM_HINTS` property. This resource determines the constraints on the window's keyboard focus. That is, it determines whether the application takes the keyboard focus away from the primary window or not. Possible values are `INPUT_APPLICATION_MODAL` and `INPUT_SYSTEM_MODAL`.

`mwmMenu` (class `MwmMenu`)

The menu items to add at the bottom of the client's window menu. The string has this format:

```
label [mnemonic] [ accelerator] mwm_f.function
```

`preeditType` (class `PreeditType`)

In Motif 1.2, specifies the input method style(s) that are available. The syntax, possible values, and default value of this resource are implementation-dependent.

`shellUnitType` (class `ShellUnitType`)

The measurement units to use in resources that specify a size or position. Possible values are `PIXELS`, `100TH_POINTS`, `100TH_MILLIMETERS`, `100TH_FONT_UNITS`, and `1000TH_INCHES`. This value is typically set by the application.

`textFontList` (class `TextFontList`)

In Motif 1.2, like the `buttonFontList` resource, but for `Text` children.

H

Obtaining Example Programs

This appendix describes how to obtain the source code for the xshowfonts program, used to create the font pictures in Appendix B. You can use the same methods to obtain other public domain clients.

In This Appendix:

FTP	899
FTPMAIL	900
BITFTP	901
UUCP	901

H

Obtaining Example Programs

The *xshowfonts* program, used to create the font pictures in Appendix B, is available electronically in a number of ways: by *ftp*, *ftpmail*, *bitftp*, and *uucp*. The cheapest, fastest, and easiest ways are listed first. If you read from the top down, the first one that works for you is probably the best. Use *ftp* if you are directly on the Internet. Use *ftpmail* if you are not on the Internet but can send and receive electronic mail to internet sites (this includes Compu-Serve users). Use BITFTP if you send electronic mail via BITNET. Use UUCP if none of the above works.

FTP

To use FTP, you need a machine with direct access to the Internet. A sample session is shown, with what you should type in boldface.

```
% ftp ftp.uu.net
Connected to ftp.uu.net.
220 FTP server (Version 6.21 Tue Mar 10 22:09:55 EST 1992) ready.
Name (ftp.uu.net:val): anonymous
331 Guest login ok, send domain style e-mail address as password.
Password: val@ora.com (use your user name and host here)
230 Guest login ok, access restrictions apply.
ftp> cd /published/oreilly/xbook/Xuser
250 CWD command successful.
ftp> binary (Very important! You must specify binary transfer for compressed files.)
200 Type set to I.
ftp> get xshowfonts.c.tar.Z
200 PORT command successful.
150 Opening BINARY mode data connection for xshowfonts.c.tar.Z.
226 Transfer complete.
ftp> quit
221 Goodbye.
%
```

If you retrieve a file from another book, it may be a compressed tar archive. In that case, extract the files from the archive by typing:

```
% zcat tar_file.c.tar.Z | tar xf -
```

System V systems require the following tar command instead:

```
% zcat tar_file.c.tar.Z | tar xof -
```

If *zcat* is not available on your system, use separate *uncompress* and *tar* commands.

If the file is a compressed shar archive, you can extract the files from the archive by typing:

```
% uncompress FILE.shar.Z
% /bin/sh FILE.shar
```

FTPMAIL

FTPMAIL is a mail server available to anyone who can send and receive electronic mail to and from Internet sites. This includes most workstations that have an email connection to the outside world, and CompuServe users. You do not need to be directly on the Internet. Here's how to do it.

You send mail to *ftpmail@decwrl.dec.com*. In the message body, give the name of the anonymous ftp host and the ftp commands you want to run. The server will run anonymous ftp for you and mail the files back to you. To get a complete help file, send a message with no subject and the single word "help" in the body. The following is an example mail session that should get you the examples. This command sends you a listing of the files in the selected directory, and the requested examples file. The listing is useful in case there's a later version of the examples you're interested in.

```
% mail ftpmail@decwrl.dec.com
Subject:
reply tim@ora.com                (where you want files mailed)
connect ftp.uu.net
chdir /published/oreilly/xbook/Xuser
dir
binary
uuencode                          (or btoa if you have it)
get xshowfonts.c.tar.Z
quit
%
```

A signature at the end of the message is acceptable as long as it appears after "quit."

All retrieved files will be split into 60KB chunks and mailed to you. You then remove the mail headers and concatenate them into one file, and then *uudecode* or *btoa* it. Once you've got the desired file, follow the directions under FTP to extract the files from the archive.

VMS, DOS, and Mac versions of *uudecode*, *btoa*, *uncompress*, and *tar* are available. The VMS versions are on *gatekeeper.dec.com* in */archive/pub/VMS*.

BITFTP

BITFTP is a mail server for BITNET users. You send it electronic mail messages requesting files, and it sends you back the files by electronic mail. BITFTP currently serves only users who send it mail from nodes that are directly on BITNET, EARN, or NetNorth. BITFTP is a public service of Princeton University. Here's how it works.

To use BITFTP, send mail containing your ftp commands to *BITFTP@PUCC*. For a complete help file, send HELP as the message body.

The following is the message body you should send to BITFTP:

```
FTP ftp.uu.net NETDATA
USER anonymous
PASS your Internet email address (not your bitnet address)
CD /published/oreilly/xbook/Xuser
DIR
BINARY
GET xshowfonts.c.tar.Z
QUIT
```

Once you've got the desired file, follow the directions under FTP to extract the files from the archive. Since you are probably not on a UNIX system, you may need to get versions of *uudecode*, *uncompress*, *btoa*, and *tar* for your system. VMS, DOS, and Mac versions are available. The VMS versions are on *gatekeeper.dec.com* in */archive/pub/VMS*.

Questions about BITFTP can be directed to Melinda Varian, *MAINT@PUCC* on BITNET.

UUCP

UUCP is standard on virtually all UNIX systems, and is available for IBM-compatible PCs and Apple Macintoshes. The examples are available by UUCP via modem from UUNET; UUNET's connect-time charges apply.

You can get the examples from UUNET whether you have an account or not. If you or your company has an account with UUNET, you will have a system with a direct UUCP connection to UUNET. Find that system, and type:

```
uucp uUNET\!~/published/oreilly/xbook/Xuser/xshowfonts.c.tar.Z\ yourhost\
!~/yourname/
```

The backslashes can be omitted if you use the Bourne shell (*sh*) instead of *csh*. The file should appear some time later (up to a day or more) in the directory */usr/spool/uucppublic/*yourname**. If you don't have an account but would like one so that you can get electronic mail, then contact UUNET at 703-204-8000.

If you don't have a UUNET account, you can set up a UUCP connection to UUNET using the phone number 1-900-468-7727. As of this writing, the cost is 50 cents per minute. The charges will appear on your next telephone bill. The login name is "uucp" with no password. For example, an *L.sys/Systems* entry might look like:

```
uunet Any ACU 19200 1-900-468-7727 ogin:--ogin: uucp
```

Your entry may vary depending on your UUCP configuration. If you have a PEP-capable modem, make sure `s50=255s111=30` is set before calling.

It's a good idea to get the file */published/oreilly/ls-lR.Z* as a short test file containing the filenames and sizes of all the files in the directory.

Once you've got the desired file, follow the directions under FTP to extract the files from the archive.

Glossary

X uses many common terms in unique ways. A good example is "children." While most, if not all, of these terms are defined where they are first used in this book, you will undoubtedly find it easier to refresh your memory by looking for them here.

Glossary

- access control list** X maintains lists of hosts that are allowed access to each server controlling a display. By default, only the local host may use the display, plus any hosts specified in the *access control list* for that display. The list is found in */etc/Xn.hosts* where *n* is the number of the display. The access control list is also known as the host access list.
- active window** The window where the input is directed. With a “pointer focus” window manager such as *twm*, you must put the pointer in a window to make it the active window. The *active window* is sometimes called the **focus window**.
- ASCII** American Standard Code for Information Interchange. This standard for data transmission assigns individual 7-bit codes to represent each of a specific set of 128 numerals, letters, and control characters.
- background color** The color that determines the backdrop of a window. For example, on monochrome displays, the root window background color is gray.
- background window** A shaded area (also called the **root window**) that covers the entire screen and upon which other windows are displayed.
- binding** An association between a function and a key and/or pointer button. *mwm* allows you to bind its functions to any key(s) on the keyboard, or to a combination of keys and pointer button (e.g., the Control key and the middle button on a 3-button pointer).
- bitmap** A grid of pixels or picture elements, each of which is white, black, or, in the case of color displays, a color. The *bitmap* client allows you to edit bitmaps, which you can use as pointers, icons, and background window patterns.
- bitmap fonts** Fonts that are pre-scaled, so that each character in each point size is stored as a bitmap. Each bitmap font requires multiple font files for storing the bitmaps in several font sizes. See also **scalable fonts**.
- border** A window can have a border that is zero or more pixels wide. If a window has a border, the border can have a solid color or a tile pattern.

client	An X application program. There are <i>client</i> programs to perform a variety of tasks, including terminal emulation and window management. Clients need not run on the same system as the display server program.
colorcell	An entry in a colormap is known as a <i>colorcell</i> . An entry contains three values specifying red, green, and blue intensities. These values are always 16-bit unsigned numbers, with zero being minimum intensity. The values are truncated or scaled by the server to match the display hardware. See also colormap .
colormap	A <i>colormap</i> consists of a set of colorcells. A pixel value indexes into the colormap to produce intensities of red, green, and blue to be displayed. Depending on hardware limitations, one or more colormaps may be installed at one time, such that windows associated with those maps display with true colors. Regardless of the number of installable colormaps, any number of virtual colormaps can be created. When needed, a virtual colormap can be installed and the existing installed colormap may have to be uninstalled. The colormap on most systems is a limited resource that should be conserved by allocating read-only colorcells whenever possible, and selecting RGB values from the predefined color database. Read-only cells may be shared between clients. See also RGB .
console window	A window that receives messages that would normally be sent to the host's console. Use the <code>-C</code> option to <i>xterm</i> to get a console window. You can also use the <i>xconsole</i> client for a read-only console window. In R5, only users who log in at the console display can get console windows.
default	A value assigned when you do not explicitly specify a different value. For example, you can specify a font to the <i>xterm</i> client using the <code>-fn</code> option or by specifying the <code>*font</code> resource. If you do not specify a font, however, the <i>xterm</i> client falls back on a 13-pixel font called <i>fixed</i> as a default font.
depth	The <i>depth</i> of a window or pixmap is the number of bits of color per pixel. A monochrome display has a depth of 1, meaning that only $2^1 = 2$ colors are available on that display: black and white. A display with 8-bit color or grayscale, meanwhile, can have $2^8 = 256$ colors or shades of gray.
device-dependent	Aspects of a system that vary depending on the hardware. For example, the number of colors available on the screen (or whether color is available at all) is a <i>device-dependent</i> feature of X.
display	A set of one or more screens driven by a single X server.
event	An action that the X server must inform clients of. For example, when the user clicks in a new window, this event is reported to the window manager so it knows to transfer focus to that window. When a window is exposed either because it is focused or because

another window is moved or exited, the associated client may have to be sent an `Expose` event so it can redraw the newly-exposed portion of the window.

- exposure** Window *exposure* occurs when a window is first mapped, or when another window that obscures it is unmapped, resized, or moved. Servers do not guarantee to preserve the contents of windows when windows are obscured or reconfigured. `Expose` events are sent to clients to inform them when contents of regions of windows have been lost and need to be regenerated.
- focus window** The window to which keyboard input is directed. By default, the keyboard focus belongs to the root window, which has the effect of sending input to whichever window has the pointer in it. Window managers in turn may enforce their own focus policy—for example, *mwm* may be configured to focus on a window only when the user explicitly clicks the pointer on it. In addition, some clients may automatically take the focus, which means they may send input to a particular window regardless of the position of the pointer.
- font** A style of text characters. Fonts and X font naming conventions are described in Chapter 6, *Font Specification*. Samples of Release 5 screen fonts are pictured in Appendix B, *Release 5 Standard Fonts*. See also **bitmap fonts** and **scalable fonts**.
- font path** When a client requests a font, servers search for that font in sequence on the font path. By default, the font path for R5 servers is set to four subdirectories of `/usr/lib/X11/fonts`: *misc*, *75dpi*, *100dpi*, and *Speedo*. You can specify an alternative font search path for the server with the *xset* client, or start the server with the `-fp` option.
- font server** The font server, new in Release 5, is a daemon that provides fonts over the network. The X server requests fonts directly from the font server instead of searching for them locally on disk.
- foreground color** The color in which text or graphics is displayed in windows and menus.
- geometry** The specification for the size and placement of a window, which can be specified with the `-geometry` option. This option takes an argument of the form: *widthxheight±xoff±yoff*.
- hexadecimal** A base-16 arithmetic system, which uses the digits A through F to represent the base-10 numbers 10 through 15. *Hexadecimal* notation (called *hex* for short) is frequently used with computers because a single hex digit can represent four binary digits (bits). X clients accept a special hexadecimal notation (prefixed by a # character) in all command-line options relating to color.
- highlighter** The horizontal band of color that moves with the pointer within a menu.

hot spot	The reference point of a pointer that corresponds to its specified position on the display. In the case of an arrow, an appropriate <i>hot spot</i> is its tip. In the case of a cross, an appropriate hot spot might be its center.
icon	A small symbol that represents a window but uses little space on the display. Converting windows to <i>icons</i> allows you to keep your display uncluttered.
input device	A hardware device that allows you to input information to the system. For a window-based system, a keyboard and pointer are the most common input devices.
keyboard focus	See focus window .
menu	A list of commands or functions, listed in a small window, which can be selected with the pointer.
modifier keys	Keys on the keyboard such as Control, Alt, and Shift. X programs recognize a set of “logical” <i>modifier key</i> functions that can be mapped to physical keys. The most frequently used of these logical keys is called the “meta” key.
mouse	An input device that, when moved across a flat surface, moves the pointer symbol correspondingly across the display. The mouse usually has buttons that can be pressed to send signals that in turn accomplish certain functions. The mouse is one type of pointer device; the representation of the mouse on the screen is also called the pointer . (See pointer .)
occluding	In a windowing system, windows may be stacked on top of each other much like a deck of cards. The window that overlays another window is said to <i>occlude</i> that window. A window need not completely conceal another window to be occluding it.
outline fonts	See scalable fonts .
padding	Space inserted to maintain alignment within the borders of windows and menus.
parameter	A value required before a client can perform a function. Also called an argument.
pixel	The smallest element of a display surface that can be addressed.
pointer	A generic name for an input device that, when moved across a flat surface, moves the pointer symbol correspondingly across the display. A <i>pointer</i> usually has buttons that can be pressed to send signals that in turn accomplish certain functions. A mouse is one type of pointer device. The pointer also refers to the symbol on your display that tracks pointer movement on your desk. Pointers allow you to make selections in menus, size and position windows and icons, and select the

window where you want to focus input. A pointer can be represented by a variety of symbols. (See **text cursor**.) Some typical X pointer symbols are the I-beam and the skull and crossbones.

- property** Windows have associated *properties*, each consisting of a name, a type, a data format, and some data. The X protocol places no interpretation on properties; they are intended as a general-purpose data storage and intercommunication mechanism for clients. There is, however, a list of predefined properties and property types so that clients can share information such as resize hints, program names, and icon formats with a window manager. In order to avoid passing arbitrary length property-name strings, each property name is associated with a corresponding integer value known as an atom.
- reverse video** Reversing the default foreground and background colors.
- RGB** An additive method for defining color in which tenths of percentages of the primaries red, green, and blue are combined to form other colors.
- root window** A shaded area (also called the **background window**) that covers the entire screen and upon which other windows are displayed.
- scalable fonts** Most fonts are pre-scaled **bitmap fonts**, meaning that each character in each point size is stored as a bitmap. Each bitmap font requires multiple font files for storing the bitmaps in several font sizes. Scalable or outline fonts are fonts that can be scaled when they are requested by a server, so only a single font file is needed to display fonts in all point sizes. An example of a scalable font is the Speedo family of fonts (distributed in */usr/lib/X11/fonts/Speedo* in X11R5). Scalable fonts can be identified by the fact that the size information in the font name is specified as "0", since they are scaled when requested.
- screen** A server may provide several independent *screens*, which may or may not have physically independent monitors. For instance, it is sometimes possible to treat a color monitor as if it were two screens, one color and one black and white.
- scrollbar** A bar on the side of a window that allows you to use the pointer to scroll up and down through the text saved in the window. For *xterm* windows, you can enable the scrollbar using the VT Options menu, using the `-sb` command-line option, or by setting the `ScrollBar` resource to `true`. The number of lines saved is usually greater than the number of lines displayed and can be controlled by the `saveLines` resource variable.
- select** A process in which you move the pointer to the desired menu item or window and click or hold down a pointer button in order to perform some action.

selection	<i>Selections</i> are a means of communication between clients using properties and events. From the user's perspective, a selection is an item of data that can be highlighted in one instance of an application and pasted into another instance of the same or a different application. The client that highlights the data is the owner, and the client into which the data is pasted is the requestor. Properties are used to store the selection data and the type of the data, while events are used to synchronize the transaction and to allow the requestor to indicate the type of data it prefers and to allow the owner to convert the data to the indicated type if possible.
server	The combination of graphics display, hardware, and X server software that provides display services for clients. The <i>server</i> also handles keyboard and pointer input. The server does not have to run on the same machine as the clients. See also X terminal .
text cursor	The standard underscore or block cursor that appears on the command line or in a text editor running an <i>xterm</i> window. To make the distinction clearer, the cursor that tracks the movement of a mouse or other pointing device is referred to as the pointer . The pointer may be associated with any number of cursor shapes, and may change shape as it moves from window to window.
tile	A pattern that is replicated (as if laying a tile) to form the background of a window or other area. This term is also used to refer to a style of window manager or application that places windows side by side instead of allowing them to overlap.
widget	A pre-defined user interface component or object. Typical widgets create graphical features such as menus, command buttons, dialog boxes, and scrollbars. Widgets make it easier to create complex applications. A common widget set also ensures a consistent user interface between applications.
window	A region on your display created by a client. For example, the <i>xterm</i> terminal emulator, the <i>xcalc</i> calculator, and the <i>bitmap</i> graphics editor all create windows. You can manipulate windows on your display using a window manager.
window manager	A client that allows you to move, resize, circulate, and iconify windows on your display.
Xcms	A device-independent color management system available in R5. The Xcms system of color management is different from the RGB system in that it does not depend on the peculiarities of the display device, but will display precisely the same color on all displays.
X terminal	A machine designed to only run an X server, with X clients running remotely on other machines. Some recent X terminals also support some local clients, such as built-in window managers.

A

acceleration, pointer, 396
access control, /etc/Xn.hosts file, 748
 user-based, 749
 xdm, 749
 XDMCP, 749
 xhost, 748
access control list, glossary definition, 899
active window, 34, 79
 glossary definition, 899
 moving focus with keystrokes, 81
aliasing font names, 144-146
alphanumeric keys, keysyms, 403
application, specifying name, 295-296
application defaults files, loading custom, 328
appres (list application resources), 25, 325
 reference page for, 444-446
arrays, converting to bitmaps, 163, 201
ASCII, glossary definition, 899
*** (asterisk) wildcard**, 308
Athena scrollbars, 97, 276
 commands, 99
Athena widget set, 15, 834-856
 class hierarchy, 834-835
 diagram, 834
 listres application, 834-835
atobm (array to bitmap converter), 24,
 163-203
 reference page for, 450-462
authorization information (xauth), 525
autorepeat option (xset), 163, 395
average width (fonts), 136

B

b option (xset), 393
background colors, 399
 glossary definition, 899
-background option (-bg) (X Toolkit), 299
background window, 7
 glossary definition, 899
 (see also root window.)
bdftopcf (font compiler), reference page for,
 447-448
bdfstosnf (font compiler), reference page for,
 449
-bd option (X Toolkit), 299
bell volume, 393
-bg option (X Toolkit), 71, 299
bindings, definition, 370
 glossary definition, 899
 tight vs. loose, 306
BITFTP, 3
bitmap, glossary definition, 899
Bitmap app-defaults file, 329
bitmap client (creating graphics), 24, 164-196
 command buttons, 164, 170, 186
 copying or moving area, 181
 copying/pasting between applications, 193
 description of, 164
 dialog boxes, 186
 drawing with, 170
 Edit menu, 191
 editing, 165
 file menu, 188
 invoking, 164, 170
 menu bar, 164
 menus, 185-196
 R5, 168
 reference page for, 450-462
 -size option, 166
 Undo command, 170
 window, 164
bitmap fonts, 126, 140
 glossary definition, 899
 size, 133
Bitmap-color file, 329
-bitmap option (xsetroot), 398-399
bitmaps, 163, 167
 converting to arrays, 163, 201

bitmaps (cont'd)
 converting to other formats, 204
 copying areas, 181
 copying/pasting between applications, 193
 creating from cursor, 201
 editing, 165, 169-185
 hot spots, 185
 image size vs. window size, 167
 inserting bitmap file into images, 190
 marking an area for editing/pasting, 178
 moving areas, 181
 portable, 204
 resizing, 165
 standard, 795-797;
 location of, 795
 (see also bitmap client.)
bmtoa (bitmap to array converter), 24,
 163-203
 reference page for, 450-462
border, glossary definition, 899
border color option (-bd), 299
bug compatibility mode, 394
button, clicking, 43
 codes, 411
 command, 18, 47, 312
 logical, 402
 pointer, 43
 pressing, 43
 push (Motif), 18
 releasing, 44
button bindings, 374-376
-bw option (X Toolkit), 298

C

c option (xset), 394
calculator (xcalc), description, 215
 function of keys, 216
 terminating, 216
Cannot allocate colormap entry, error mes-
 sage, 358
Can't Open display, error message, 65
Can't open error file, error message, 157
cascading menus, 378
character cell fonts, 129
character set, 136
check boxes (Motif), 275
-chime option, 71
class, definition, 307
 Gadget (Motif), 857
 hierarchy; (see widget class hierarchy)
 Object, 827-828

 RectObj, 827-828
 resource names, 307
clicking buttons, 43
click-to-type focus, (see explicit focus)
clients, application name, 295-296
 customizing, 25, 72-74, 303
 desk accessories, 210
 display manager, 23
 display options, 63
 glossary definition, 900
 location of default values, 304
 Motif, 265-290
 options, 59, 63
 removing, 231
 running on another machine, 63
 setup, 391-412
 specifying default characteristics for, 72
 standard vs. Motif, 15-19
 starting additional, 58-68
 user-contributed, 241-259
 window manager, 22
client-specific resources (mwm), 380
clipboard, 315
 (see also xclipboard.)
CLIPBOARD selection, 107, 315-319
 (see also xclipboard.)
clock, 211
 (see also oclock; xclock.)
Clock-color app-defaults file, 329
color, background (glossary definition), 899
 databases; alternative, 348;
 changing, 356;
 editing and compiling, 357;
 fixing corrupted, 358
 definition, 397
 determining number available, 354
 device-independent; (see Xcms)
 displaying, 354
 editing, 245-259
 for screen elements, 346
 foreground (glossary definition), 901
 graphics; (see pixmap)
 names, 343-345;
 adding, 356
 previewing, 348
 previewing RGB, 242
 problems allocating, 359
 reverse video (glossary definition), 903
 RGB, 343-345, 351-353
 RGB (glossary definition), 903
 RGB; available colors, 345-349
 root window, 399
 screen; setting resources for, 328

- color** (cont'd)
 - specifying, 343-361;
 - as hexadecimal numbers, 345;
 - root window (xsetroot), 399
 - values, finding, 356
 - Xcms, 343-345, 349-350, 353-354
- colorcell**, definition, 354
 - glossary definition, 900
 - read-only, 355
 - read/write, 355, 397
 - shared, 355
- colormap**, 236, 397
 - description, 354
 - glossary definition, 900
- command box (Motif)**, 285-287
- command button widget (Athena)**, 18, 188
- command-line options**, 59, 71-72, 293-299, 303
 - b (xset), 393
 - background, 299
 - bd (border color), 299
 - bg (background), 71, 299
 - bitmap (xsetroot), 398
 - border color, 299
 - borderwidth, 298
 - bw (border width), 298
 - c (xset), 394
 - def (xsetroot), 398
 - display, 63-67
 - edit (xrdb), 324
 - fg, 71
 - fn (font), 298
 - fn (xfd), 801
 - foreground, 299
 - fp (font path), 395
 - geometry, 167
 - grammar (xmodmap), 408
 - gray/grey (xsetroot), 398
 - hd, 71
 - help (xsetroot), 398
 - hl, 71
 - iconic, 297
 - led (xset), 395
 - list of standard, 293
 - mag (xmag), 197
 - merge (xrdb), 324
 - mod (xsetroot), 398
 - n (xmodmap), 408
 - name, 296, 322
 - pke (xmodmap), 405
 - pp (xmodmap), 411
 - r (xset), 395
 - reverse, 298
 - s (xset), 396
 - sb, 93
 - sh (bitmap), 167
 - size (bitmap), 166
 - solid (xsetroot), 398
 - sw (bitmap), 167
 - title, 295-296
 - update, 71
 - verbose (xmodmap), 408
 - xrm, 321
 - xset, 393
- commands**, bitmap editing, 169-185
 - for terminating xterm window, 116
 - Main Options menu (xterm), 115
 - pointer, 169
 - Tek Options menu (xterm), 121
 - text editing widget, 227
 - UNIX, running in temporary xterm, 96
 - VT Options menu (xterm), 118
- component appearance resources (mwm)**, 380
- Composite widget class**, 829-830
- config file (font server)**, 156
- console messages**, 750
 - clearing, 89, 210
- console window**, glossary definition, 900
- Constraint widget class**, 829
- Control key**, 227, 314
- conventions of this book**, v
- copying**, images, with bitmap, 181, 193
 - images, with xmag, 199
 - text in windows, 100-106
- Core widget class**, 828-829
 - resources, 828-829
- cursor**, cursor font, 801
 - standard, 801-803
- curves**, drawing with bitmap, 170
- customization resource**, 304, 329
- customizing**, clients, 25, 72-74, 303
 - keyboard, 401
 - mwm, 365-388;
 - icon box, 385;
 - .mwmrc file, 365, 485;
 - system.mwmrc file, 368-376;
 - .xresources file, 365
 - pointer, 401
 - when to do, 391
 - X environment, 72-74
- cut buffer strings**, 100
 - vs. selections, 106-112
- cutting**, images, with bitmap, 181, 193

D

database, resource, 323
DCC file, 360
DEC VT102, (see VT102)
default, glossary definition, 900
defaults, setting, 304-305, 310
-def option (xsetroot), 398
deiconifying windows, 46
Delete key, 228
depth, glossary definition, 900
desk accessories, 24, 210-228
Device Color Characterization file, 360
device-dependent, glossary definition, 900
dialog boxes (Athena), 187
 bitmap, 186
dialog boxes (Motif), 279-287, 381
directories, font, 140
display, depth of, 354
DISPLAY environment variable, 63, 64, 67, 745
display, fonts, 127, 129, 148;
 (see also xfd.)
 glossary definition, 900
 information, generating, 240;
 (see also xdpynfo.)
 manager, 23;
 (see also xdm.)
 name; where stored, 64
 server, 21
 setting, 393-397
 setting after remote login, 67
-display option (X Toolkit), 63-67
DNDDemo program (Motif), 290
drag and drop (Motif), 289-290
drawn buttons (Motif), 273-274
dump file, (see window dump file)

E

Edit menu (bitmap), 191
-edit option (xrdb), 324
editres, 25, 331-340
 Commands menu, 333
 editres protocol, 332
 incompatible clients, 332
 Tree menu, 333
environment variables, DISPLAY, 67, 89, 239, 745
 TERMCAP, 95-96
 XENVIRONMENT, 327

error messages, Cannot allocate colormap entry, 358
 Can't Open display, 65
 Can't open error file, 157
 X Error of failed request, 158
escape sequences, 807-815
event, definition, 21, 312
 glossary definition, 900
 input, 312
 translations, 312, 312-319, 313, 817-821
 types, 817
examples, obtaining electronically, 1
exiting, xmag, 201
 xterm window, 56
explicit focus, 10-12, 78-79
exposure event, glossary definition, 901

F

feedback boxes, 381
-fg option (X Toolkit), 71, 299
File menu (bitmap), 188
file selection dialog (Motif), 283-285
files, .mwmrc, 26
 resource, 303
 .Xauthority, 749
 .Xdefaults, 115
 .xinitrc, 304, 744
 .Xresources, 113, 115, 304
 .xsession, 304
Flood Fill (bitmap), 175
f.menu function, 371
-fn option (xfd), 801
focus, definition, 12
 moving with keystrokes, 81
 policies; explicit, 10-12, 78-79;
 pointer, 10-12;
 setting, 384
 restoring, 82
 window, 79;
focus window, glossary definition, 901
focusAutoRaise, 79
font path option (xset), 395
font paths, changing, 395
font server, 126, 155-160
 glossary definition, 901
 information (fsinfo), 127, 159
fonts, 75-dpi vs 100-dpi, 134-135
 additional style, 136
 aliases, 144, 146
 average width, 136
 bitmap, 126, 140

fonts (cont'd)

- bold and demi-bold, 133
- character cell, 129, 136
- character set, 136
- check compiled font (showfont), 127, 158-160
- choosing the easy way, 127
- conventions in book, v
- databases, creating (mkfontdir), 144
- default, 127
- directories, 140
- display specified (xshowfonts), 127, 129
- displaying (xfont), 125, 127, 146, 148
- dump contents of (showfont), 127, 158-160
- elements of font names, 138
- families, 129, 140
- font server (fs), 126, 155-160
- fonts.dir files, 143
- foundries, 135
- glossary definition, 901
- italic vs. oblique, 132
- list available (xlsfonts), 127, 158-160
- list from a server (fslsfonts), 127, 158-160
- monospace, 129, 136
- naming conventions, 128
- number of fonts available, 129
- on command-line, 298
- outline, 126, 140;
 - (see also scalable fonts.)
- pictures of, 753-791
- point size, 133
- Portable Compiled Format, 126
- previewing (xfontsel), 127, 148
- printer, 125
- proportional, 129, 136
- R5, 126
- reverse italic and reverse oblique, 133
- scalable outline, 126, 140;
 - (see also scalable fonts.)
- screen, 125, 298
- search path, 140-141
- selecting (xfontsel), 127, 148
- serif and sans-serif, 132, 136
- server information (fsinfo), 127, 159
- set width, 136
- slant, 132
- suitable for xterm, 127
- weight, 132
- wildcarding, 137-139
- xterm, 130
- fonts.alias files**, 144
- fonts.dir files (font databases)**, 143-144
- foreground**, colors, 399

- foreground option (-fg) (X Toolkit)**, 299
- foundries (fonts)**, 135
- f.post_wmnu function**, 371
- frame**, mwm, 30
- frame option (xwd)**, 229
- fs (font server)**, 24, 126, 155-160
 - reference page for, 471-474
- f.separator function**, 371
- f.set_behavior function**, 366
- fsinfo (display font server info)**, 127, 159
 - reference page for, 475-476
- fslsfonts (list fonts from a server)**, 24, 127, 158-160
 - reference page for, 477-478
- fstobdf (convert font server to bdf)**, reference page for, 479
- FTPMAIL**, 2
- functions**, mwm, 371

G

- gadgets**, ArrowButton (Motif), 858
 - CascadeButton (Motif), 859-860
 - Gadget (Motif), 857, 867-868
 - Label (Motif), 868-869
 - PushButton (Motif), 877-878
 - Separator (Motif), 888
 - ToggleButton (Motif), 892-893
- geometry**, glossary definition, 901
- geometry option (X Toolkit)**, 59-63
 - and bitmap, 167
- grammar option (xmodmap)**, 408
- graphics**, creating with bitmap, 164-196
 - magnifying with xmag, 196
 - utilities, 163-206
- graymap**, converting to another format, 204
 - portable, 204
- gray/grey option (xsetroot)**, 398
- grayscale graphics**, (see graymap)
- grip**, definition, 224
- GUI (graphical user interface)**, i, 16
- Gumby bitmap**, 166, 203
- hd option (xclock)**, 71
- help option (xsetroot)**, 398

H

- hexadecimal color specification**, 348
 - glossary definition, 901
- highlighter**, glossary definition, 901
- hl option (xclock)**, 71
- hot spot**, glossary definition, 902

hot spots (bitmap), 185

I

ICCCM, 22

icon, glossary definition, 902

icon box, 385-388

pack, 90

-iconic option (X Toolkit), 297

iconifying windows, 45, 56

icons, converting to windows, 46, 87

definition, 9

focusing input on, 80

managing, 86-88

managing with Window Menu, 87

moving, 51

raising, 49

reorganizing, 90

starting window as, 297

Window Menu actions on, 87

input, events, 312

focus, 34, 79

focusing on an icon, 80

input device, glossary definition, 902

instance, definition, 307

resource names, 307

interactivePlacement resource (mwm), 42

ISO Latin-1 character set, 136

K

keyboard, autorepeat, 395

bell, 393

customization, 401

focus, 34

mappings; changing, 401-412

modifiers, 817

preferences, 393-397

keyclick volume, 394

keycode, 403

keys, Control, 227, 314

Delete, 228

determining default mappings, 405-406

mapping, 401-412;

example, 409

Meta, 78, 227, 314, 405

modifier, 78-82, 401, 401-412

keysym, 403

definition, 403

determining, 406

mapping, 408

values, 819

killing, clients, 59, 214, 231

oclock, 232

server, 231

windows, 55-56, 83, 116

xclock, 232

L

led option (xset), 395

LEDs, 395

lines, drawing with bitmap, 171

list fonts from a server (fslsfonts), 127,

158-160

list fonts (xlsfonts), 127, 158-160

(see also xlsfonts.)

listres (list resources for widgets), 834-835

reference page for, 480-481

load average, 67

logging in, 23, 30-33

logical, font convention, 128

keyname, 78, 404

pointer button, 402

loose bindings (resources), 306, 310

M

magic cookie, 748-749

magnifying screen, 196

(see also xmag.)

-mag option (xmag), 197

Main Options menu (xterm), 112-117

commands, 115-116

mode toggles, 115

managing icons, 86-88

mapping, changing key, 391

definition, 312

event-action, 314

modifier keys, 401, 404, 411

possibilities with xmodmap, 408

translation table, 313

Maximize command button, 47-48

menu bars (Motif), 267-269

menus, bitmap, 185-196

cascading, 378

editres, 333

glossary definition, 902

Main Options (xterm), 112-117

mwm (window manager), 113, 371

option (Motif), 269-270

pop-up (Motif), 269

pull-down (Motif), 267-269

Root Menu (mwm), 88-90

- menus (cont'd)**
 - submenus, 378
 - tear-off (Motif), 270-272
 - Tek Options (xterm), 93, 120-121
 - types (Motif), 267-272
 - VT Fonts (xterm), 119
 - VT Options (xterm), 96, 117-119
 - Window Menu (mwm), 55-56, 83-88
 - xterm, 112-121
 - merge option (xrdb)**, 324
 - Meta key**, 78, 227-228, 314, 405
 - Minimize command button**, 45-47
 - mkfontdir (create font databases)**, 144
 - reference page for, 482-483
 - mode toggles**, Main Options menu (xterm), 115
 - Tek Options menu (xterm), 121
 - VT Options menu (xterm), 118
 - modifier keys**, 78-82, 401
 - glossary definition, 902
 - mapping, 401-412
 - mod option (xsetroot)**, 398-399
 - monochrome screen**, setting resources for, 328
 - monospaced fonts**, 129
 - Motif**, 15, 265-290
 - applications, 15-19
 - widget set, 15
 - window manager, 77, 77-90
(see also mwm.)
 - Motif class hierarchy**, 857-858
 - diagram, 857
 - Motif toolkit**, check boxes, 275
 - command box, 285-287
 - dialog boxes, 279-287
 - drag and drop, 289-290
 - drawn buttons, 273-274
 - file selection dialog, 283-285
 - menus, 267-272
 - periodic table of widgets, 265-266
 - prompt dialog, 281
 - push buttons, 272-274
 - radio boxes, 274-275
 - scrollbars, 276-277
 - selection dialog, 282-283
 - text windows, 277-279
 - Motif widget set**, 857-895
 - mouse**, glossary definition, 902
 - mouse option (xset)**, 396
 - mouse tracking**, 813
 - moving**, icons, 51
 - windows, 50
 - mre (Motif resource editor)**, 16-17
 - mwm (Motif window manager)**, 8, 22, 33, 44-52, 77-90
 - activating changes to, 90, 366
 - background processes, 35
 - bringing up, 34
 - client-specific resources, 383, 503-507
 - component appearance resources, 381, 493-494
 - customizing, 365-388
 - environment variables, 508
 - focus window; changing, 81; selecting, 79
 - frame; features, 82; title area, 50
 - functions, 371;
 - f.menu, 371;
 - f.post_wmenu, 371;
 - f.separator, 371;
 - f.set_behavior, 366
 - menus, 113;
 - cascading, 378;
 - creating new, 378;
 - specifications, 371
 - .mwmrc file, 26, 485-492
 - mwm-specific appearance/behavior resources, 380, 494-503
 - overview, 77
 - reference page for, 484-508
 - resources, 493-507
 - restarting, 90, 366
 - stacking order, changing, 80
 - starting, 34-35
 - typing in a window, 35
 - .mwmrc file**, 26, 485-492
 - mwm-specific resources**, 380, 494-503
 - name option (X Toolkit)**, 296, 322
- N**
- naming conventions**, fonts, 128
 - network**, running client over, 63
 - New Window command**, mwm Root Menu, 89
 - n option (xmodmap)**, 408
- O**
- Object class**, 827-828
 - objects**, Sme (Athena), 849-850
 - SmeBSB (Athena), 850-851
 - SmeLine (Athena), 851
 - occluded**, glossary definition, 902
 - oclock (analog clock)**, 58, 211-214

- app-defaults file, 329
- killing, 214, 232
- reference page for, 509-511
- removing, 59, 214
- old-rgb.txt file, 348
- OPEN LOOK, 16
- option menus (Motif), 269-270
- options, 293-299
 - client, 59
 - command-line; (see command-line options)
- OSF/Motif window manager, 77, 77-90
 - (see also mwm.)
- outline fonts, 126, 140
 - size, 133
 - (see also scalable fonts.)

P

- Pack Icons command, mwm Root Menu, 90
- padding, glossary definition, 902
- parameter, glossary definition, 902
- pasting, images, with bitmap, 181, 193
 - images, with xmag, 199
 - text, 100-106, 105
- path, including X in, 743
- PBM Toolkit, 163, 204
- periodic table (Motif), 265-266
- pipes and pointer interaction, 230
- pixels, 60
 - glossary definition, 902
- pixmap, converting to another format, 204
 - portable, 204
- pke option (xmodmap), 405
- point size, 133
- pointer, acceleration, 396
 - changing root window, 400
 - commands, 169
 - customization, 401
 - definition, 9
 - dragging item with, 44
 - glossary definition, 902
 - mapping, 411
 - possible cursor images, 801
 - using, 43-44
- pointer button, 402, 411
 - mappings, changing, 391
 - (see also button.)
- pointer focus, 10-12
- pop-up menus (Motif), 269
- Portable Bitmap Toolkit, 163, 204
- Portable Compiled Format (fonts), 126
- postscript translation (xpr), 229-230

- pp option (xmodmap), 411
- PRIMARY selection, 100
- printer fonts, (see fonts)
- printing utilities, 229-230
- programs, obtaining electronically, 1
- prompt dialog (Motif), 281
- property, 903
- proportional fonts, 129
- pull-down menus (Motif), 18, 267-269
- push button widget (Motif), 18, 272-274

Q

- ? (question mark) wildcard, 308
- Quit command (xterm Main Options menu), 117

R

- R5, fonts, pictures of, 753
- radio boxes (Motif), 274-275
- raveling.txt file, 348
- rbg.pag file, 346
- read-only colorcell, 355, 397
- read/write colorcell, 355, 397
- rectangles, drawing with bitmap, 174
- RectObj class, 827-828
- redrawing, windows, 117
- Refresh command, mwm Root Menu, 89, 210
- refreshing the screen, 750
- remote system, logging in to, 67
 - monitoring load on, 67
 - running client on, 63
- resize (reset terminal window), 95
 - reference page for, 512-513
- resizing windows, 95-96
 - using pointer, 52
- resource database manager, 26, 304, 320
 - (see also xrdb.)
- RESOURCE_MANAGER property, 323
- resources, 62, 72
 - ArrowButtonGadget, 858
 - ArrowButton widget, 858
 - AsciiText widget, 853
 - Box widget, 836
 - BulletinBoard widget, 859
 - CascadeButton widget, 860
 - CascadeButtonGadget, 860
 - class names of, 307
 - client-specific, 380, 383
 - color vs. monochrome screens, 328
 - Command widget, 836, 860

resources (cont'd)

- component appearance (mwm), 380-381
 - Dialog widget, 837
 - DialogShell widget, 861
 - Display widget, 862
 - DrawingArea widget, 862
 - DrawnButton widget, 863
 - event translations of, 312-319
 - files of, 303
 - FileSelectionBox widget, 863
 - Form widget, 838, 864
 - Frame widget, 866
 - Gadget, 867
 - Grip widget, 839
 - instance names, 307
 - Label widget, 839, 868
 - LabelGadget, 868
 - list of common, 311
 - List widget, 840, 870
 - listing current, 325
 - loading, 320, 366
 - MainWindow widget, 871
 - management, 303, 323-340
 - Manager widget, 872
 - MenuShell widget, 873
 - MessageBox widget, 874
 - mwm-specific, 380, 382
 - Paned widget, 842
 - PanedWindow widget, 875
 - Panner widget, 844
 - Porthole widget, 846
 - precedence rules, 309
 - Primitive widget, 876
 - PushButton widget, 878
 - PushButtonGadget, 878
 - removing definitions, 325
 - Repeater widget, 846
 - sample file, 320
 - Scale widget, 882
 - Screen widget, 883
 - Scrollbar widget, 847
 - ScrollBar widget, 885
 - ScrolledWindow widget, 886
 - SelectionBox widget, 887
 - Separator widget, 888
 - SeparatorGadget, 888
 - setting, 73-74, 319-322;
 - screen-specific, 330;
 - mwm, 380;
 - with xrdb, 323
 - Simple widget, 848
 - SimpleMenu widget, 849
 - SmeBSB object, 850
 - SmeLine object, 851
 - specification of, 303-340
 - from command line, 321
 - StripChart widget, 852
 - testing and editing, 331-340
 - Text widget, 853, 889
 - TextField widget, 891
 - tight vs. loose bindings, 310
 - Toggle widget, 854
 - ToggleButton widget, 893
 - ToggleButtonGadget, 893
 - translation table, 312
 - Tree widget, 855
 - VendorShell widget, 894
 - Viewport widget, 856
 - widget, 879
- Restart**, mwm Root Menu, 90
 - restarting mwm**, 90
 - reverse video**, 298, 399
 - glossary definition, 903
 - reverse option (X Toolkit)**, 298
 - RGB**, glossary definition, 903
 - RGB color model**, 343, 351-353
 - available colors, 345-349
 - fixing corrupted database, 358
 - specifying as hexadecimal numbers, 348
 - triplet, 352
 - rgb.dir file**, 346
 - rgb.txt file**, 346
 - and RGB triplets, 352
 - display colors, 242, 245, 348
 - rlogin**, setting DISPLAY, 67
 - Root Menu (mwm)**, 88-90, 89, 371
 - customizing, 376
 - root window**, changing pointer, 400
 - characteristics; setting, 398
 - color, 399
 - definition, 7
 - glossary definition, 903
 - setting (xsetroot), 398
 - rsh command**, 65
 - rv option (X Toolkit)**, 298
 - sb option (xterm)**, 93

S

- scalable fonts**, 126, 140
 - glossary definition, 903
- screen**, clear messages obscuring, 89, 210
 - fonts; (see fonts)
 - glossary definition, 903
 - magnifying, 196

screen (cont'd)
 resolution, 134-135
 saver option (xset), 396
 savers, 396

scripts, startup, 744-747

scrollbar, glossary definition, 903

Scrollbar widget, Athena, 18, 222
 Motif, 18, 222

scrollbars, and xterm, 93
 Athena, 97, 99, 276
 creating in xterm window, 96
 Motif, 276-277
 running xterm with, 96
 using, 98
 VT Options menu (xterm), 96

ScrolledWindow widget (Motif), 222

search path, including X in, 743
 setting, 33

security, /etc/Xn.hosts file, 748
 magic cookie, 749
 restricting access to servers, 748
 user-based access, 749
 xdm, 749
 XDMCP, 749
 xhost, 748

select, glossary definition, 903

selection dialog (Motif), 282-283

selections, copying, 100-106
 glossary definition, 904
 manipulating, 106-112
 pasting, 105
 saving multiple, 107
 text, 100
 vs. cut buffers, 106-112

Send CONT signal command (xterm Main Options menu), 116

Send HUP Signal command (xterm Main Options menu), 116

Send INT Signal command (xterm Main Options menu), 116

Send KILL Signal command (xterm Main Options menu), 117

Send STOP signal command (xterm Main Options menu), 116

Send TERM Signal command (xterm Main Options menu), 116

server, closing connection, 231
 control access, 748-749
 definition, 21
 display, 21
 glossary definition, 904
 reference page for, 437-443
 restricting access to, 748
 starting, 29, 744

sessreg (manage utmp/wtmp entries), reference page for, 514-515

set width (fonts), 136

setup clients, 391-412

-sh option, bitmap, 167

shell environment variables, (see environment variables)

shell scripts, (see scripts)

Shell widget class, 829-830

showfont (display font data), 127, 158-160
 reference page for, 516-517

showrgb (display rgb color database), 347
 reference page for, 518

showsfnf (display compiled font), reference page for, 519

Shuffle Down command (mwm Root Menu), 89

Shuffle Up command (mwm Root Menu), 89

-size option (bitmap), 166"

-solid option (xsetroot), 398-399

spacing (fonts), 136

Speedo font directory, 126

stacking order, changing with keystrokes, 80

standard bitmaps, 795-797

standard cursors, 801-803

starting X, 29-37
 bringing up window manager, 34
 manually, 33
 setting search path, 743
 with xinit, 33

startup scripts, 745

submenus (mwm), 378

-sw option, bitmap, 167

system management, 743-750

system.mwmrc file, 365, 368-376

T

tear-off menus (Motif), 270-272

Tek Options menu (xterm), 112
 commands, 121
 description of items, 120-121
 mode toggles, 121

Tektronix 4014, 23
 control sequences

telnet, setting DISPLAY, 67

temporary xterm windows, running commands
 in, 96

termcap, 94-95

TERMCAP environment variable, 95-96

terminal, emulator, 6;

terminal, control sequences (cont'd)
 definition, 23;
 see also xterm, 93
 type; xterm, 94-95
 windows, resetting, 95

terminating xterm window, 56, 116, 297

terminfo, 94-95

text, copying and pasting, 100-106
 selecting for copying, 101
 selections; extending, 102-103;
 large, 111

text cursor, glossary definition, 904

Text widget (Athena), 222, 227

text windows (Motif), 277-279

thomas.txt file, 348

tight bindings (resources), 306, 310

tile, 904

title area, description, 8
 in xterm window, 30
 mwm frame, 50

titlebar, 45
 and moving windows, 50
 description, 8-9

-title option (X Toolkit), 295-296

toggle buttons (Motif), 274-275

Toolkit options, (see command-line options)

toolkits, definition, 304

translation table, definition, 312
 example, 314
 mappings in, 313
 syntax, 314, 817-821

translations, definition, 312

twm (tab window manager), 8

U

-update option (xclock), 71

user-contributed clients, 241-259

V

variables, environment;(see environment variables)
 resource; (see resources)

-verbose option (xmodmap), 408

vertical panes, 224

Viewport widget (Athena), 222

viewres (view widget tree), reference page for, 520-524

VPaned widget (Athena), 224

VT Fonts menu (xterm), 94, 112
 description of items, 119

VT Options menu (xterm), 112
 Allow 80/132 Column Switching, 118
 description of items, 117-119
 mode toggles, 118

VT102 (xterm), 23
 control sequences, 808
 VT Options menu, 117-119

W

widget, glossary definition, 904

widget class hierarchy, 827-830
 Composite, 829-830
 Constraint, 829
 Core, 828-829
 Object, 827-828
 RectObj, 827-828
 Shell, 829-830

widgets, ArrowButton (Motif), 858
 Athena, 15, 834-856
 attributes, 306
 binding; loose vs. tight, 306
 Box (Athena), 836
 BulletinBoard (Motif), 858-859
 callback, 827
 CascadeButton (Motif), 859-860
 Command (Athena), 18, 188, 830, 832, 836-837
 Command (Motif), 860-861
 Composite; (see Composite widget class)
 Constraint; (see Constraint widget class)
 Core; (see Core widget class)
 defining conventions, 306
 definition, 209
 Dialog (Athena), 187, 837
 DialogShell (Motif), 861
 Display (Motif), 861-862
 DrawingArea (Motif), 862
 DrawnButton (Motif), 863
 FileSelectionBox (Motif), 863-864
 Form (Athena), 829, 832, 837-839
 Form (Motif), 864-866
 Frame (Motif), 866-867
 Grip (Athena), 839
 hierarchy, 305;
 (see widget class hierarchy)
 inheriting resources, 832-834
 instance names, 832
 introduction, 827
 Label (Athena), 830, 839-840
 Label (Motif), 868-869
 List (Athena), 840-841

widgets (cont'd)

- List (Motif), 869-871
- MainWindow (Motif), 871
- Manager (Motif), 871-873
- MenuButton (Athena), 841-842
- MenuShell (Motif), 873
- MessageBox (Motif), 873-874
- Motif, 15, 857-895
- OPEN LOOK, 16
- Paned (Athena), 842-844
- PanedWindow (Motif), 874-876
- Panner (Athena), 844-845
- Porthole (Athena), 845-846
- Primitive (Motif), 876-877
- PushButton (Motif), 18, 877-878
- relationship between widgets, 832-834
- Repeater (Athena), 846
- RowColumn (Motif), 878-881
- Scale (Motif), 881-883
- Screen (Motif), 883-884
- Scrollbar (Athena), 97, 222, 846-848
- ScrollBar (Motif), 222, 884-885
- ScrolledWindow (Motif), 222, 885-886
- SelectionBox (Motif), 886-888
- Separator (Motif), 888
- Shell; (see Shell widget class)
- Simple (Athena), 830, 848
- SimpleMenu (Athena), 848-849
- StripChart (Athena), 851-852
- subclassing, 827
- Text (Athena), 222, 227, 852-854
- Text (Motif), 888-891
- TextField (Motif), 891-892
- Toggle (Athena), 854
- ToggleButton (Motif), 892-893
- Tree (Athena), 855
- using in an application, 830-832
- VendorShell (Motif), 894-895
- Viewport (Athena), 222, 856
- VPaned (Athena), 224

wildcards, * (asterisk), 308

- ? (question mark), 308
- in font names, 137-139
- in resource specifications, 308

window dump file, creating (xwd), 229

- displaying (xwud), 229
- printing, 229-230
- undumping (xwud), 230

window managers, 22

- definition, 7
- glossary definition, 904
- steps for starting, 34
- (see also twm, mwm.)

Window Menu (mwm), 371

- command button, 55, 83
- displaying, 84
- managing icons with, 87
- reasons for using, 85
- removing menu, 85
- selecting items, 86

windows, active, 79

- closing, 56
- creating, 34, 41-56
- definition, 7
- deiconifying, 46, 87
- displaying information about, 233
- enlargening, 47
- exiting xterm, 56
- focus, 79
- geometry, 59-63
- glossary definition, 904
- hierarchy of, 236
- icon box, 385
- iconifying, 45, 56
- killing; with Window Menu button, 55, 83;
 - with xkill, 231
- managing with mwm frame, 82
- maximizing, 47
- minimizing, 47
- moving, 50
- multiple, 43
- placing using pointer, 42
- raising, 49
- redrawing, 117
- removing, 59, 214
- resizing, 52, 95-96
- root, 7, 398
- running programs in temporary xterm, 96
- size and location, 59-63
- starting as icon, 297
- switching between, 42
- Tektronix, 93
- terminating, 116
- title, 295-296
- typing in, 35
- vertically tiled, 224
- width of, 298

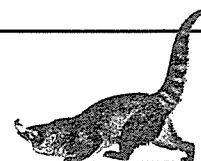
X

- X color management system**, (see Xcms)
- X Error of failed request**, error message, 158
- X terminal**, glossary definition, 904
- X Window System**, description of, 5
 - display server, 21

- X Window System** (cont'd)
 - displays, 6-15
 - environment; customizing, 72-74
 - getting started, 29-37
 - including in search path, 743
 - overview of architecture, 19
 - reference page for, 419-436
 - terminals, 6
 - X Toolkit, 15
- xauth (display authorization information)**, reference page for, 525-528
- .Xauthority file**, 749
- xbiff (mail notification)**, 24, 216
 - reference page for, 529-531
- xcalc (calculator)**, 24, 215, 215-216, 216
 - reference page for, 532-538
- xclipboard (save text selections)**, 17, 106-112, 107, 222, 315
 - command buttons and functions, 109
 - reference page for, 539-541
- xclock (analog or digital clock)**, 24, 71, 211-214
 - killing, 214
 - reference page for, 542-545
 - removing, 214
- Xcms**, 344, 349-350, 353-354
 - advantages of, 344
 - color spaces, 349
 - database, creating, 359
 - device-specific tuning, 360
 - glossary definition, 904
- xcmsdb (edit screen color properties)**, reference page for, 546-547
- xcol (display/change colors)**, 242, 348
 - reference page for, 548-550
- xcoloredit (edit RGB colors)**, 245, 348, 350, 356
 - reference page for, 551-552
- xconsole (console message monitor)**, reference page for, 553-555
- xcrta (create Xcms database)**, reference page for, 556-557
- .Xdefaults file**, 115, 304
 - vs. xrdb, 320
 - .Xresources file; (see also .Xresources file.) 72
- xditview (display ditroff DVI files)**, reference page for, 558-560
- xdm (display manager)**, 23
 - access control features, 749
 - and starting X, 29
 - login window, 30
 - reference page for, 561-578
- xdpr (window dump to printer)**, 230
 - reference page for, 579-580
- xdpyinfo (list display information)**, 25, 240
 - reference page for, 581-584
- xedit (text editor)**, 112, 222
 - reference page for, 585-595
- XENVIRONMENT environment variable**, 327
- xev (track events)**, 406
 - reference page for, 596-597
- xfd (font displayer)**, 24, 126-127, 146, 230
 - fn option, 801
 - reference page for, 598-600
- xfontsel (preview and select font)**, 24, 126-127, 148-153
 - reference page for, 601-604
- xhost (control access to server)**, 748
 - reference page for, 605-606
- xinit (start X server)**, 33, 744
 - reference page for, 607-609
- .xinitrc file**, 744
- xkill (remove window)**, 25, 231-233
 - reference page for, 610-611
- xload (poll system load average)**, 24, 67, 217
 - reference page for, 612-614
- xlogo (display X logo)**, reference page for, 615-616
- xlsatoms (list atoms on server)**, reference page for, 617
- xlsclients (list running clients)**, 25, 239-240
 - reference page for, 618
- xlsfonts (list available fonts)**, 24, 126-128, 158-160
 - reference page for, 619-620
- xlswins (list window tree)**, 236
 - reference page for, 621-622
- xmag (magnify screen portion)**, 24, 163, 196-201
 - command buttons, 198
 - copying and pasting with, 199
 - description of, 196
 - quitting, 201
 - R5, 196, 198
 - reference page for, 623-626
 - selecting area to magnify, 197
- xman (display manual pages)**, 24, 218
 - as a Viewport, 222
 - exiting, 222
 - reference page for, 627-634
- xmh (message handling system)**, reference page for, 635-659



More Titles from O'Reilly

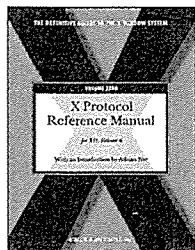


X Window System Programming

Volume 0: X Protocol Reference Manual

VOLUME
0

Edited by Adrian Nye
4th Edition January 1995
458 pages, ISBN 1-56592-083-X



This manual describes the X Network Protocol, which underlies all software for Version 11 of the X Window System. It not only provides a practical demonstration of what is involved in a client session, but also an extensive set of reference pages for each protocol request and event. Reference pages, alphabetized for easy access, include encoding of requests and replies.

The fourth edition of *X Protocol Reference Manual* includes protocol clarifications of X11 Release 6 and can be used with any release of X. Note: This edition does not contain the Inter-Client Communication Conventions Manual (ICCCM) or the X Logical Font Description Convention (XLFD). This material will be included in an upcoming O'Reilly book.

Volume 1: Xlib Programming Manual

VOLUME
1

By Adrian Nye
3rd Edition July 1992
824 pages, ISBN 1-56592-002-3

Covering X11 Release 5, the *Xlib Programming Manual* is a complete guide to programming the X library (Xlib), the lowest level of programming interface to X. It includes introductions to internationalization, device-independent color, font service, and scalable fonts.

Includes chapters on:

- X Window System concepts
- A simple client application
- Window attributes
- The graphics context
- Graphics in practice
- Color
- Events
- Interclient communication
- Internationalization
- The Resource Manager
- A complete client application
- Window management
- Other programming techniques

This manual is a companion to Volume 2, *Xlib Reference Manual*.

Volume 2: Xlib Reference Manual

VOLUME
2

By Adrian Nye
3rd Edition June 1992
1138 pages, ISBN 1-56592-006-6

Volume 2, *Xlib Reference Manual*, is a complete programmer's reference for Xlib. Covers X11 Release 4 and Release 5.

Contents Include:

- Reference pages for Xlib functions
- Reference pages for event types
- Permuted index to Xlib functions
- Description of macros and reference pages for their function versions
- Listing of the server-side color database
- Alphabetical index and description of structures
- Alphabetical index and description of defined symbols
- KeySyms and their meaning
- Illustration of the standard cursor font
- Function group index to the right routine for a particular task
- Reference pages for Xlib-related Xmu functions (miscellaneous utilities)
- Four single-page reference aids for the GC and window attributes
- Index

New features in the third edition include:

- Over 100 new man pages covering Xcms, internationalization, and the function versions of macros.
- Updating to the R5 spec.
- New "Returns" sections on all the functions which return values, making this information easier to find.

Volume 4M: X Toolkit Intrinsic Programming Manual

VOLUME
4M

Motif Edition, By Adrian Nye & Tim O'Reilly
2nd Edition August 1992, 674 pages
ISBN 1-56592-013-9

A complete guide to programming with the X Toolkit Intrinsic, the library of C language routines that facilitates the design of user interfaces with reusable components called widgets. This book provides concepts and examples that show how to use the various X Toolkit routines. The first few chapters are devoted to using widgets; the remainder of the book covers the more complex task of writing new widgets.

Uses the Motif 1.2 widget set in examples and covers X11 Release 5.

O'REILLY™

TO ORDER: 800-998-9938 • order@oreilly.com • <http://www.oreilly.com/>

OUR PRODUCTS ARE AVAILABLE AT A BOOKSTORE OR SOFTWARE STORE NEAR YOU.

FOR INFORMATION: 800-998-9938 • 707-829-0515 • info@oreilly.com

X Window System Programming

Volume 5: X Toolkit Intrinsic Reference Manual



Edited by David Flanagan
3rd Edition April 1992
916 pages, ISBN 1-56592-007-4

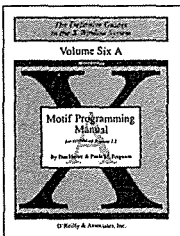
The X Toolkit Intrinsic Reference Manual is a complete programmer's reference for the X Toolkit. This volume is based on Xt documentation from the X Consortium and has been reorganized and expanded for X11 Release 5. It provides reference pages for each of the Xt functions, as well as the widget classes defined by Xt and the Athena widgets, and many useful appendices.

This manual is a companion to Volume 4M, X Toolkit Intrinsic Programming Manual.

Volume 6A: Motif Programming Manual



By Dan Heller, Paula Ferguson & David Brennan
2nd Edition February 1994
1016 pages, ISBN 1-56592-016-3



The Motif Programming Manual is a source for complete, accurate, and insightful guidance on Motif application programming. There is no other book that covers the ground as thoroughly or as well as this one.

The Motif Programming Manual describes how to write applications using the Motif toolkit from the Open Software Foundation

(OSF). The book goes into detail on every Motif widget class, with useful examples that will help programmers to develop their own code. Anyone doing Motif programming who doesn't want to have to figure it out alone needs this book.

In addition to information on Motif, the book is full of tips about programming in general and about user interface design. It includes a tutorial on UIL; coverage of drag-and-drop, tear-off menus, and internationalization as implemented in the Motif widgets such as Text and TextField; plus the entire book has been checked for accuracy with Motif 1.2 (while remaining usable with Motif 1.1). Complements Volume 6B, Motif Reference Manual.

Volume 6B: Motif Reference Manual



By Paula Ferguson & David Brennan
1st Edition June 1993
920 pages, ISBN 1-56592-038-4

The *Motif Reference Manual* is a complete programmer's reference for the Motif toolkit from the Open Software Foundation (OSF). Motif has become the standard user interface for X Window System applications, and the Motif toolkit makes it easy for programmers to build applications that conform with the Motif "look and feel."

This book provides reference pages for the Motif functions and macros, the Motif and Xt widget classes, the Mrm functions, the Motif clients, and the UIL file format, data types, and functions. The reference material has been expanded from the appendices of the first edition of Volume 6 and covers Motif 1.2. This book is designed to be used with Volume 6A, *Motif Programming Manual*, which describes how to build applications using the Motif toolkit and provides a complete tutorial with programming examples.

O'REILLY™

TO ORDER: 800-998-9938 • order@oreilly.com • <http://www.oreilly.com/>

OUR PRODUCTS ARE AVAILABLE AT A BOOKSTORE OR SOFTWARE STORE NEAR YOU.

FOR INFORMATION: 800-998-9938 • 707-829-0515 • info@oreilly.com

How to stay in touch with O'Reilly

1. Visit Our Award-Winning Web Site

<http://www.oreilly.com/>

- ★ "Top 100 Sites on the Web" — *PC Magazine*
- ★ "Top 5% Web sites" — *Point Communications*
- ★ "3-Star site" — *The McKinley Group*

Our web site contains a library of comprehensive product information (including book excerpts and tables of contents), downloadable software, background articles, interviews with technology leaders, links to relevant sites, book cover art, and more. File us in your Bookmarks or Hotlist!

2. Join Our Email Mailing Lists

New Product Releases

To receive automatic email with brief descriptions of all new O'Reilly products as they are released, send email to: listproc@online.oreilly.com

Put the following information in the first line of your message (*not* in the Subject field):
subscribe oreilly-news

O'Reilly Events

If you'd also like us to send information about trade show events, special promotions, and other O'Reilly events, send email to:

listproc@online.oreilly.com

Put the following information in the first line of your message (*not* in the Subject field):
subscribe oreilly-events

3. Get Examples from Our Books via FTP

There are two ways to access an archive of example files from our books:

Regular FTP

- ftp to:
[ftp.oreilly.com](ftp://ftp.oreilly.com)
(login: anonymous
password: your email address)
- Point your web browser to:
<ftp://ftp.oreilly.com/>

FTPMAIL

- Send an email message to:
ftpmail@online.oreilly.com
(Write "help" in the message body)

4. Contact Us via Email

order@oreilly.com

To place a book or software order online. Good for North American and international customers.

subscriptions@oreilly.com

To place an order for any of our newsletters or periodicals.

books@oreilly.com

General questions about any of our books.

software@oreilly.com

For general questions and product information about our software. Check out O'Reilly Software Online at <http://software.oreilly.com/> for software and technical support information. Registered O'Reilly software users send your questions to: website-support@oreilly.com

cs@oreilly.com

For answers to problems regarding your order or our products.

booktech@oreilly.com

For book content technical questions or corrections.

proposals@oreilly.com

To submit new book or software proposals to our editors and product managers.

international@oreilly.com

For information about our international distributors or translation queries. For a list of our distributors outside of North America check out:
<http://www.oreilly.com/www/order/country.html>

O'Reilly & Associates, Inc.

101 Morris Street, Sebastopol, CA 95472 USA

TEL 707-829-0515 or 800-998-9938

(6am to 5pm PST)

FAX 707-829-0104

O'REILLY™

TO ORDER: 800-998-9938 • order@oreilly.com • <http://www.oreilly.com/>

OUR PRODUCTS ARE AVAILABLE AT A BOOKSTORE OR SOFTWARE STORE NEAR YOU.

FOR INFORMATION: 800-998-9938 • 707-829-0515 • info@oreilly.com

International Distributors

UK, EUROPE, MIDDLE EAST AND NORTHERN AFRICA (EXCEPT FRANCE, GERMANY, SWITZERLAND, & AUSTRIA)

INQUIRIES

International Thomson Publishing Europe
Berkshire House
168-173 High Holborn
London WC1V 7AA
United Kingdom
Telephone: 44-171-497-1422
Fax: 44-171-497-1426
Email: itpint@itps.co.uk

ORDERS

International Thomson Publishing Services,
Ltd.
Cheriton House, North Way
Andover, Hampshire SP10 5BE
United Kingdom
Telephone: 44-264-342-832 (UK)
Telephone: 44-264-342-806 (outside UK)
Fax: 44-264-364418 (UK)
Fax: 44-264-342761 (outside UK)
UK & Eire orders: itpuk@itps.co.uk
International orders: itpint@itps.co.uk

FRANCE

Editions Eyrolles
61 bd Saint-Germain
75240 Paris Cedex 05
France
Fax: 33-01-44-41-11-44

FRENCH LANGUAGE BOOKS

All countries except Canada
Telephone: 33-01-44-41-46-16
Email: geodif@eyrolles.com
English language books
Telephone: 33-01-44-41-11-87
Email: distribution@eyrolles.com

GERMANY, SWITZERLAND, AND AUSTRIA

INQUIRIES

O'Reilly Verlag
Balthasarstr. 81
D-50670 Köln
Germany
Telephone: 49-221-97-31-60-0
Fax: 49-221-97-31-60-8
Email: anfragen@oreilly.de

ORDERS

International Thomson Publishing
Königswinterer Straße 418
53227 Bonn, Germany
Telephone: 49-228-97024 0
Fax: 49-228-441342
Email: order@oreilly.de

JAPAN

O'Reilly Japan, Inc.
Kiyoshige Building 2F
12-Banchi, Sanei-cho
Shinjuku-ku
Tokyo 160-0008 Japan
Telephone: 81-3-3356-5227
Fax: 81-3-3356-5261
Email: kenji@oreilly.com

INDIA

Computer Bookshop (India) PVT. Ltd.
190 Dr. D.N. Road, Fort
Bombay 400 001 India
Telephone: 91-22-207-0989
Fax: 91-22-262-3551
Email: cbsbom@giasbm01.vsnl.net.in

HONG KONG

City Discount Subscription Service Ltd.
Unit D, 3rd Floor, Yan's Tower
27 Wong Chuk Hang Road
Aberdeen, Hong Kong
Telephone: 852-2580-3539
Fax: 852-2580-6463
Email: citydis@ppn.com.hk

KOREA

Hanbit Media, Inc.
Sonyoung Bldg. 202
Yeksam-dong 736-36
Kangnam-ku
Seoul, Korea
Telephone: 822-554-9610
Fax: 822-556-0363
Email: hant93@chollian.dacom.co.kr

SINGAPORE, MALAYSIA, AND THAILAND

Addison Wesley Longman Singapore PTE
Ltd.
25 First Lok Yang Road
Singapore 629734
Telephone: 65-268-2666
Fax: 65-268-7023
Email: daniel@longman.com.sg

PHILIPPINES

Mutual Books, Inc.
429-D Shaw Boulevard
Mandaluyong City, Metro
Manila, Philippines
Telephone: 632-725-7538
Fax: 632-721-3056
Email: mbikikog@mnl.sequel.net

CHINA

Ron's DataCom Co., Ltd.
79 Dongwu Avenue
Dongxihu District
Wuhan 430040
China
Telephone: 86-27-3892568
Fax: 86-27-3222108
Email: hongfeng@public.wh.hb.cn

ALL OTHER ASIAN COUNTRIES

O'Reilly & Associates, Inc.
101 Morris Street
Sebastopol, CA 95472 USA
Telephone: 707-829-0515
Fax: 707-829-0104
Email: order@oreilly.com

AUSTRALIA

Woodslane Pty. Ltd.
7/5 Vuko Place, Warriewood NSW 2102
P.O. Box 935
Mona Vale NSW 2103
Australia
Telephone: 61-2-9970-5111
Fax: 61-2-9970-5002
Email: info@woodslane.com.au

NEW ZEALAND

Woodslane New Zealand Ltd.
21 Cooks Street (P.O. Box 575)
Waganui, New Zealand
Telephone: 64-6-347-6543
Fax: 64-6-345-4840
Email: info@woodslane.com.au

THE AMERICAS

McGraw-Hill Interamericana Editores,
S.A. de C.V.
Cedro No. 512
Col. Atlampa 06450
Mexico, D.F.
Telephone: 52-5-541-3155
Fax: 52-5-541-4913
Email: mcgraw-hill@infosel.net.mx

SOUTH AFRICA

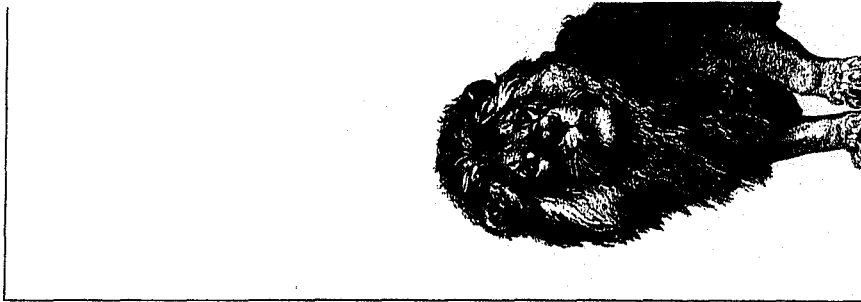
International Thomson Publishing
South Africa
Building 18, Constantia Park
138 Sixteenth Road
P.O. Box 2459
Halfway House, 1685 South Africa
Telephone: 27-11-805-4819
Fax: 27-11-805-3648

O'REILLY™

TO ORDER: 800-998-9938 • order@oreilly.com • <http://www.oreilly.com/>

OUR PRODUCTS ARE AVAILABLE AT A BOOKSTORE OR SOFTWARE STORE NEAR YOU.

FOR INFORMATION: 800-998-9938 • 707-829-0515 • info@oreilly.com



O'REILLY™

O'Reilly & Associates, Inc.
101 Morris Street
Sebastopol, CA 95472-9902
1-800-998-9938

Visit us online at:
<http://www.ora.com/>
orders@ora.com



O'REILLY WOULD LIKE TO HEAR FROM YOU

Which book did this card come from?

Where did you buy this book?

- Bookstore
- Direct from O'Reilly
- Bundled with hardware/software
- Other _____
- Computer Store
- Class/seminar

What operating system do you use?

- UNIX
- Windows NT
- Other _____
- Macintosh
- PC(Windows/DOS)

What is your job description?

- System Administrator
- Network Administrator
- Web Developer
- Other _____
- Programmer
- Educator/Teacher

Please send me O'Reilly's catalog, containing a complete listing of O'Reilly books and software.

Name _____

Company/Organization _____

Address _____

City _____

State _____

Zip/Postal Code _____

Country _____

Telephone _____

0926

Internet or other email address (specify network) _____

Nineteenth century wood engraving
of a bear from the O'Reilly &
Associates Nutshell Handbook®
Using & Managing UUCP.



PLACE
STAMP
HERE

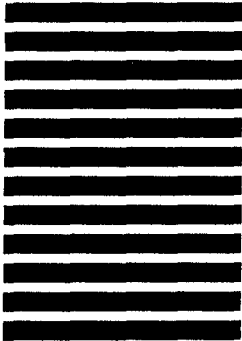


NO POSTAGE
NECESSARY IF
MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS MAIL PERMIT NO. 80 SEBASTOPOL, CA

Postage will be paid by addressee

O'Reilly & Associates, Inc.
101 Morris Street
Sebastopol, CA 95472-9902



Motif Edition

X Window System User's Guide

This book orients the new user to window system concepts and provides detailed tutorials for many client programs, including the *xterm* terminal emulator and the *mwm* window manager. "A Quick Start Approach to X" in the Preface guides you through the most fundamental topics, which are covered in Part One of the guide. Once you have a basic knowledge of the system, the later chapters explain how to customize the X environment and provide sample configurations.

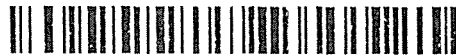
The *X Window System User's Guide, Motif Edition*, reflects X11 Release 5 and Motif 1.2. Though Motif is not strictly part of the X Window System, but a commercial product layered on top of it, it has gained wide acceptance. The book describes how to use the Motif *mwm* window manager in conjunction with the standard MIT X clients. It also describes differences between these clients (built with the MIT Athena widget set) and commercial client programs built with the OSF/Motif widget set.

The guide describes:

- Starting the system and opening the first client windows
- Using the *xterm* terminal emulator and the *mwm* window manager
- Most standard release clients, including programs for graphics, printing, font manipulation, window/display information and removing the windows, as well as several "desktop" utilities
- New R5 features including scalable outline fonts, the font server program, and device-independent color
- Customizing the window manager, keyboard, display, and certain basic features of any client program

The books in the X Window System Series are based in part on the original MIT X Window System documentation, but are far more comprehensive, easy to use, and are loaded with examples, tutorials, and helpful hints. Over 20 major computer manufacturers recommend or license volumes in the series. In short, these books are the definitive guides to the X Window System.

ISBN 1-56592-045-5 US \$34.95
CAN \$49.95



X000FIUHTR

X USERS GUIDE MOTIF R5... V. 3 (X WINDOW SYSTEM)
Used, Very Good

9 "781565"9201--

Visit O'Reilly
on the Web at
www.oreilly.com



Printed on Recycled Paper

O'Reilly & Associates, Inc.