NAME

xterm − terminal emulator for X

SYNOPSIS

xterm [-*toolkitoption* ...] [-*option* ...] [*shell*]

DESCRIPTION

The *xterm* program is a terminal emulator for the X Window System. It provides DEC VT102/VT220 (VTxxx) and Tektronix 4014 compatible terminals for programs that cannot use the window system directly. If the underlying operating system supports terminal resizing capabilities (for example, the SIGWINCH signal in systems derived from 4.3bsd), *xterm* will use the facilities to notify programs running in the window whenever it is resized.

The VTxxx 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 may 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 VTxxx window, and the ''Tek Options'' menu in the 4014 window.

EMULATIONS

The VT102 emulation is fairly complete, but does not support autorepeat. Double-size characters are displayed properly if your font server supports scalable fonts. The VT220 emulation does not support soft fonts, it is otherwise complete. *Termcap*(5) entries that work with *xterm* include an optional platform-specific entry, ''xterm,'' ''vt102,'' ''vt100'' and ''ansi,'' and ''dumb.'' *xterm* automatically searches the termcap file in this order for these entries and then sets the ''TERM'' and the ''TERMCAP'' environment variables. You may also use ''vt220,'' but must set the terminal emulation level with the **decTerminalID** resource. (The ''TERMCAP'' environment variable is not set if *xterm* is linked against a terminfo library, since the requisite information is not provided by the termcap emulation of terminfo libraries).

Many of the special *xterm* features may be modified under program control through a set of escape sequences different from the standard VT102 escape sequences. (See the *Xterm Control Sequences* document.)

The Tektronix 4014 emulation is also fairly good. It supports 12-bit graphics addressing, scaled to the window size. Four different font sizes and five different lines types are supported. There is no write-through or defocused mode support. 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 ''**COPY***yyyy*-*MM*-*dd*.*hh*:*mm*:*ss*'', where *yyyy*, *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 *xterm* is started in, or the home directory for a login *xterm*).

Not all of the features described in this manual are necessarily available in this version of *xterm*. Some (e.g., the non-VT220 extensions) are available only if they were compiled in, though the most commonly-used are in the default configuration.

OTHER FEATURES

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*(1) to switch to the alternate screen for editing and to restore the screen on exit. A popup menu entry makes it simple to switch between the normal and alternate screens for cut and paste.

In either VT102 or Tektronix mode, there are escape sequences to change the name of the windows. Additionally, in VT102 mode, *xterm* implements the window-manipulation control sequences from *dtterm*, such as resizing the window, setting its location on the screen.

Xterm allows character-based applications to receive mouse events (currently button-press and release events, and button-motion events) as keyboard control sequences. See *Xterm Control Sequences* for details.

OPTIONS

The *xterm* terminal emulator accepts the standard X Toolkit command line options as well as many application-specific options. If the option begins with a '+' instead of a '-', the option is restored to its default value. The **-version** and **-help** options are interpreted even if *xterm* cannot open the display, and are useful for testing and configuration scripts:

-version This causes *xterm* to print a version number to the standard output.

-help This causes *xterm* to print out a verbose message describing its options, one per line. The message is written to the standard output. *Xterm* generates this message, sorting it and noting whether a "**-option**" or a "**+option**" turns the feature on or off, since some features historically have been one or the other. *Xterm* generates a concise help message (multiple options per line) when an unknown option is used, e.g.,

xterm -z

If the logic for a particular option such as logging is not compiled into *xterm*, the help text for that option also is not displayed by the **-help** option.

One parameter (after all options) may be given. That overrides *xterm*'s built-in choice of shell program. Normally *xterm* checks the SHELL variable. If that is not set, *xterm* tries to use the shell program specified in the password file. If that is not set, *xterm* uses */bin/sh*. If the parameter names an executable file, *xterm* uses that instead. The parameter must be an absolute path, or name a file found on the user's PATH (and thereby construct an absolute path). The **-e** option cannot be used with this parameter since it uses all parameters following the option.

The other options are used to control the appearance and behavior. Not all options are necessarily configured into your copy of *xterm*:

-132 Normally, the VT102 DECCOLM escape sequence that switches between 80 and 132 column mode is ignored. This option causes the DECCOLM escape sequence to be recognized, and the *xterm* window will resize appropriately.

-ah This option indicates that *xterm* should always highlight the text cursor. By default, *xterm* will display a hollow text cursor whenever the focus is lost or the pointer leaves the window.

+ah This option indicates that *xterm* should do text cursor highlighting based on focus.

-ai This option disables active icon support if that feature was compiled into *xterm*. This is equivalent to setting the *vt100* resource **activeIcon** to ''false''.

+ai This option enables active icon support if that feature was compiled into *xterm*. This is equivalent to setting the *vt100* resource **activeIcon** to ''true''.

-aw This option indicates that auto-wraparound should be allowed. This allows the cursor to automatically wrap to the beginning of the next line when it is at the rightmost position of a line and text is output.

+aw This option indicates that auto-wraparound should not be allowed.

-b *number*

This option specifies the size of the inner border (the distance between the outer edge of the characters and the window border) in pixels. That is the *vt100 internalBorder* resource. The default is 2.

+bc turn off text cursor blinking. This overrides the **cursorBlink** resource.

-bc turn on text cursor blinking. This overrides the **cursorBlink** resource.

-bcf *milliseconds*
set the amount of time text cursor is off when blinking via the *cursorOffTime* resource.

-bcn *milliseconds*
set the amount of time text cursor is on when blinking via the *cursorOffTime* resource.

-bdc Set the *vt100* resource **colorBDMode** to ‘‘false’’, disabling the display of characters with bold attribute as color

+bdc Set the *vt100* resource **colorBDMode** to ‘‘true’’, enabling the display of characters with bold attribute as color rather than bold

-cb Set the *vt100* resource **cutToBeginningOfLine** to ‘‘false’’.

+cb Set the *vt100* resource **cutToBeginningOfLine** to ‘‘true’’.

-cc *characterclassrange***:***value***[,...]**
This sets classes indicated by the given ranges for using in selecting by words. See the section specifying character classes. and discussion of the *charClass* resource.

-cjk_width
Set the **cjkWidth** resource to ‘‘true’’. When turned on, characters with East Asian Ambiguous (A) category in UTR 11 have a column width of 2. Otherwise, they have a column width of 1. This may be useful for some legacy CJK text terminal-based programs assuming box drawings and others to have a column width of 2. It also should be turned on when you specify a TrueType CJK double-width (bi-width/monospace) font either with **-fa** at the command line or **faceName** resource. The default is ‘‘false’’

+cjk_width
Reset the **cjkWidth** resource.

-class *string*
This option allows you to override *xterm*’s resource class. Normally it is ‘‘XTerm’’, but can be set to another class such as ‘‘UXTerm’’ to override selected resources.

-cm This option disables recognition of ANSI color-change escape sequences. It sets the *colorMode* resource to ‘‘false’’.

+cm This option enables recognition of ANSI color-change escape sequences. This is the same as the *vt100* resource **colorMode**.

-cn This option indicates that newlines should not be cut in line-mode selections. It sets the *cutNewline* resource to ‘‘false’’.

+cn This option indicates that newlines should be cut in line-mode selections. It sets the *cutNewline* resource to ‘‘true’’.

-cr *color*
This option specifies the color to use for text cursor. The default is to use the same foreground color that is used for text. It sets the *cursorColor* resource according to the parameter.

-cu This option indicates that *xterm* should work around a bug in the *more*(1) program that causes it to incorrectly display lines that are exactly the width of the window and are followed by a line beginning with a tab (the leading tabs are not displayed). This option is so named because it was originally thought to be a bug in the *curses*(3x) cursor motion package.

+cu This option indicates that *xterm* should not work around the *more*(1) bug mentioned above.

-dc This option disables the escape sequence to change dynamic colors: the vt100 foreground and background colors, its text cursor color, the pointer cursor foreground and background colors, the Tektronix emulator foreground and background colors, its text cursor color and highlight color. The option sets the *dynamicColors* option to ‘‘false’’.

+dc This option enables the escape sequence to change dynamic colors. The option sets the *dynamicColors* option to ‘‘true’’.

-e *program* **[** *arguments* **. . .]**
This option specifies the program (and its command line arguments) to be run in the *xterm* window. It also sets the window title and icon name to be the basename of the program being executed if neither *-T* nor *-n* are given on the command line. **This must be the last option on the command line.**

-en *encoding*
This option determines the encoding on which *xterm* runs. It sets the **locale** resource. Encodings other than UTF-8 are supported by using *luit*. The **-lc** option should be used instead of **-en** for systems with locale support.

-fb *font* This option specifies a font to be used when displaying bold text. 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 will be used as the normal font and the bold font will be produced by overstriking this font. The default is to do overstriking of the normal font. See also the discussion of **boldFont**, **boldMode** and **alwaysBoldMode** resources.

-fa *pattern*
This option sets the pattern for fonts selected from the FreeType library if support for that library was compiled into *xterm*. This corresponds to the **faceName** resource. When a CJK double-width font is specified, you also need to turn on the **cjkWidth** resource. See also the **renderFont** resource, which combines with this to determine whether FreeType fonts are initially active.

-fbb This option indicates that *xterm* should compare normal and bold fonts bounding boxes to ensure they are compatible. It sets the **freeBoldBox** resource to ‘‘false’’.

+fbb This option indicates that *xterm* should not compare normal and bold fonts bounding boxes to ensure they are compatible. It sets the **freeBoldBox** resource to ‘‘true’’.

-fbx This option indicates that *xterm* should not assume that the normal and bold fonts have VT100 line-drawing characters. If any are missing, *xterm* will draw the characters directly. It sets the **forceBoxChars** resource to ‘‘false’’.

+fbx This option indicates that *xterm* should assume that the normal and bold fonts have VT100 line-drawing characters. It sets the **forceBoxChars** resource to ‘‘true’’.

-fd *pattern*
This option sets the pattern for double-width fonts selected from the FreeType library if support for that library was compiled into *xterm*. This corresponds to the **faceNameDoublesize** resource.

-fi *font* This option sets the font for active icons if that feature was compiled into *xterm*. See also the discussion of the **iconFont** resource.

-fs *size* This option sets the pointsize for fonts selected from the FreeType library if support for that library was compiled into *xterm*. This corresponds to the **faceSize** resource.

-fw *font* This option specifies the font to be used for displaying wide text. By default, it will attempt to use a font twice as wide as the font that will be used to draw normal text. If no doublewidth font is found, it will improvise, by stretching the normal font. This corresponds to the **wideFont** resource.

-fwb *font*
This option specifies the font to be used for displaying bold wide text. By default, it will attempt to use a font twice as wide as the font that will be used to draw bold text. If no doublewidth font is found, it will improvise, by stretching the bold font. This corresponds to the **wideBoldFont** resource.

-fx *font* This option specifies the font to be used for displaying the preedit string in the "OverTheSpot" input method. See also the discussion of the **ximFont** resource.

-hc *color*
(see **-selbg**).

-hf This option indicates that HP Function Key escape codes should be generated for function keys. It sets the **hpFunctionKeys** resource to ‘‘true’’.

+hf This option indicates that HP Function Key escape codes should not be generated for function keys. It sets the **hpFunctionKeys** resource to ‘‘false’’.

-hm Tells *xterm* to use **highlightTextColor** and **highlightColor** to override the reversed foreground/background colors in a selection. It sets the **highlightColorMode** resource to ‘‘true’’.

+hm Tells *xterm* not to use **highlightTextColor** and **highlightColor** to override the reversed foreground/background colors in a selection. It sets the **highlightColorMode** resource to ‘‘false’’.

-hold Turn on the **hold** resource, i.e., *xterm* will not immediately destroy its window when the shell command completes. It will wait until you use the window manager to destroy/kill the window, or if you use the menu entries that send a signal, e.g., HUP or KILL.

+hold Turn off the **hold** resource, i.e., *xterm* will immediately destroy its window when the shell command completes.

-ie Turn on the **ptyInitialErase** resource, i.e., use the pseudo-terminal’s sense of the stty erase value.

+ie Turn off the **ptyInitialErase** resource, i.e., set the stty erase value using the **kb** string from the termcap entry as a reference, if available.

-im Turn on the **useInsertMode** resource, which forces use of insert mode by adding appropriate entries to the TERMCAP environment variable.

+im Turn off the **useInsertMode** resource.

-into *windowId*
Given an X window identifier (a decimal integer), *xterm* will reparent its top-level shell widget to that window. This is used to embed *xterm* within other applications.

-j This option indicates that *xterm* should do jump scrolling. It corresponds to the **jumpScroll** resource. Normally, text is scrolled one line at a time; this option allows *xterm* to move multiple lines at a time so that it does not fall as far behind. Its use 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 as well as the ‘‘VT Options’’ menu can be used to turn this feature on or off.

+j This option indicates that *xterm* should not do jump scrolling.

-k8 This option sets the **allowC1Printable** resource. When **allowC1Printable** is set, *xterm* overrides the mapping of C1 control characters (code 128-159) to treat them as printable.

+k8 This option resets the **allowC1Printable** resource.

-kt *keyboardtype*
This option sets the **keyboardType** resource. Possible values include: ‘‘unknown’’, ‘‘default’’, ‘‘hp’’, ‘‘sco’’, ‘‘sun’’, ‘‘tcap’’ and ‘‘vt220’’.

The value ‘‘unknown’’, causes the corresponding resource to be ignored.

The value ‘‘default’’, suppresses the associated resources **hpFunctionKeys**, **scoFunctionKeys**, **sunFunctionKeys, tcapFunctionKeys** and **sunKeyboard**, using the Sun/PC keyboard layout.

-l Turn logging on. Normally logging is not supported, due to security concerns. Some versions of *xterm* may have logging enabled. The logfile is written to the directory from which *xterm* is invoked. The filename is generated, of the form

XtermLog.*XXXXXX*

or

Xterm.log.*hostname.yyyy.mm.dd.hh.mm.ss.XXXXXX*

depending on how *xterm* was built.

+l Turn logging off.

-lc Turn on support of various encodings according to the users' locale setting, i.e., LC_ALL, LC_CTYPE, or LANG environment variables. This is achieved by turning on UTF-8 mode and by invoking *luit* for conversion between locale encodings and UTF-8. (*luit* is not invoked in UTF-8 locales.) This corresponds to the **locale** resource.

The actual list of encodings which are supported is determined by *luit*. Consult the *luit* manual page for further details. See also the discussion of the **-u8** option which supports UTF-8 locales.

+lc Turn off support of automatic selection of locale encodings. Conventional 8bit mode or, in UTF-8 locales or with **-u8** option, UTF-8 mode will be used.

-lcc *path*
File name for the encoding converter from/to locale encodings and UTF-8 which is used with **-lc** option or **locale** resource. This corresponds to the **localeFilter** resource.

-leftbar Force scrollbar to the left side of VT100 screen. This is the default, unless you have set the rightScrollBar resource.

-lf *filename*
Specify the log-filename. See the **-l** option.

-ls This option indicates that the shell that is started in the *xterm* window will 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).

The **-ls** flag and the **loginShell** resource are ignored if **-e** is also given, because *xterm* does not know how to make the shell start the given command after whatever it does when it is a login shell - the user's shell of choice need not be a Bourne shell after all. Also, *xterm -e* is supposed to provide a consistent functionality for other applications that need to start text-mode programs in a window, and if **loginShell** were not ignored, the result of ~/.profile might interfere with that.

If you do want the effect of **-ls** and **-e** simultaneously, you may get away with something like
xterm -e /bin/bash -l -c "my command here"

Finally, **-ls** is not completely ignored, because *xterm -ls -e* does write a */etc/wtmp* entry (if configured to do so), whereas *xterm -e* does not.

+ls This option indicates that the shell that is started should not be a login shell (i.e., it will be a normal ''subshell'').

-mb This option indicates that *xterm* should ring a margin bell when the user types near the right end of a line. This option can be turned on and off from the ''VT Options'' menu.

+mb This option indicates that margin bell should not be rung.

-mc *milliseconds*
This option specifies the maximum time between multi-click selections.

-mesg Turn off the **messages** resource, i.e., disallow write access to the terminal.

+mesg Turn on the **messages** resource, i.e., allow write access to the terminal.

-mk_width
Set the **mkWidth** resource to ''true''. This makes *xterm* use a built-in version of the wide-character width calculation. The default is ''false''

+mk_width
Reset the **mkWidth** resource.

-ms *color*
This option specifies the color to be used for the pointer cursor. The default is to use the foreground color. This sets the *pointerColor* resource.

-nb *number*
This option specifies the number of characters from the right end of a line at which the margin bell, if enabled, will ring. The default is 10.

-nul This option disables the display of underlining.

+nul This option enables the display of underlining.

-pc This option enables the PC-style use of bold colors (see boldColors resource).

+pc This option disables the PC-style use of bold colors.

-pob This option indicates that the window should be raised whenever a Control-G is received.

+pob This option indicates that the window should not be raised whenever a Control-G is received.

-rightbar
Force scrollbar to the right side of VT100 screen.

-rvc This option disables the display of characters with reverse attribute as color.

+rvc This option enables the display of characters with reverse attribute as color.

-rw This option indicates that reverse-wraparound should be allowed. This allows the cursor to back up from the leftmost column of one line to the rightmost column of the previous line. This is very useful for editing long shell command lines and is encouraged. This option can be turned on and off from the ''VT Options'' menu.

+rw This option indicates that reverse-wraparound should not be allowed.

-s This option indicates that *xterm* may scroll asynchronously, meaning that the screen does not have to be kept completely up to date while scrolling. This allows *xterm* to run faster when network latencies are very high and is typically useful when running across a very large internet or many gateways.

+s This option indicates that *xterm* should scroll synchronously.

-samename
Does not send title and icon name change requests when the request would have no effect: the name is not changed. This has the advantage of preventing flicker and the disadvantage of requiring an extra round trip to the server to find out the previous value. In practice this should never be a problem.

+samename
Always send title and icon name change requests.

-sb This option 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 so that those lines can be viewed. This option may be turned on and off from the ''VT Options'' menu.

+sb This option indicates that a scrollbar should not be displayed.

-selbg *color*
This option specifies the color to use for the background of selected text. If not specified, reverse video is used. See the discussion of the **highlightColor** resource.

-selfg *color*
This option specifies the color to use for selected text. If not specified, reverse video is used. See the discussion of the **highlightTextColor** resource.

-sf This option indicates that Sun Function Key escape codes should be generated for function keys.

+sf This option indicates that the standard escape codes should be generated for function keys.

-si This option indicates that output to a window should not automatically reposition the screen to the bottom of the scrolling region. This option can be turned on and off from the ''VT Options'' menu.

+si This option indicates that output to a window should cause it to scroll to the bottom.

-sk This option indicates that pressing a key while using the scrollbar to review previous lines of text should cause the window to be repositioned automatically in the normal position at the bottom of the scroll region.

+sk This option indicates that pressing a key while using the scrollbar should not cause the window to be repositioned.

-sl *number*
This option specifies the number of lines to save that have been scrolled off the top of the screen. This corresponds to the **saveLines** resource. The default is 64.

-sm This option, corresponding to the **sessionMgt** resource, indicates that *xterm* should set up session manager callbacks.

+sm This option indicates that *xterm* should not set up session manager callbacks.

-sp This option indicates that Sun/PC keyboard should be assumed, providing mapping for keypad '+' to ',', and CTRL-F1 to F13, CTRL-F2 to F14, etc.

+sp This option indicates that the standard escape codes should be generated for keypad and function keys.

-t This option indicates that *xterm* should start in Tektronix mode, rather than in VT102 mode. Switching between the two windows is done using the ''Options'' menus. *Termcap*(5) entries that work with *xterm* ''tek4014,'' ''tek4015,'' ''tek4012'', ''tek4013'' and ''tek4010,'' and ''dumb.'' *xterm* automatically searches the termcap file in this order for these entries and then sets the ''TERM'' and the ''TERMCAP'' environment variables.

+t This option indicates that *xterm* should start in VT102 mode.

-tb This option, corresponding to the **toolBar** resource, indicates that *xterm* should display a toolbar (or menubar) at the top of its window. The buttons in the toolbar correspond to the popup menus, e.g., control/left/mouse for "Main Options".

+tb This option indicates that *xterm* should not set up a toolbar.

-ti *term_id*
Specify the name used by *xterm* to select the correct response to terminal ID queries. It also specifies the emulation level, used to determine the type of response to a DA control sequence. Valid values include vt52, vt100, vt101, vt102, and vt220 (the "vt" is optional). The default is vt100. The term_id argument specifies the terminal ID to use. (This is the same as the **decTerminalID** resource).

-tm *string*
This option specifies a series of terminal setting keywords followed by the characters that should be bound to those functions, similar to the *stty* program. The keywords and their values are described in detail in the **ttyModes** resource.

-tn *name*
This option specifies the name of the terminal type to be set in the TERM environment variable. It corresponds to the **termName** resource. This terminal type must exist in the terminal database (termcap or terminfo, depending on how *xterm* is built) and should have *li#* and *co#* entries. If the terminal type is not found, *xterm* uses the built-in list ''xterm'', ''vt102'', etc.

-u8 This option sets the **utf8** resource. When **utf8** is set, *xterm* interprets incoming data as UTF-8. This sets the **wideChars** resource as a side-effect, but the UTF-8 mode set by this option prevents it from being turned off. If you must turn it on and off, use the **wideChars** resource.

This option and the **utf8** resource are overridden by the **-lc** and **-en** options and **locale** resource. That is, if *xterm* has been compiled to support *luit*, and the **locale** resource is not ''false'' this option is ignored. We recommend using the **-lc** option or the ''**locale: true**'' resource in UTF-8 locales when your operating system supports locale, or **-en UTF-8** option or the

''**locale: UTF-8**'' resource when your operating system does not support locale.

+u8 This option resets the **utf8** resource.

-ulc This option disables the display of characters with underline attribute as color rather than with underlining.

+ulc This option enables the display of characters with underline attribute as color rather than with underlining.

-ulit This option, corresponding to the **italicULMode** resource, disables the display of characters with underline attribute as italics rather than with underlining.

+ulit This option, corresponding to the **italicULMode** resource, enables the display of characters with underline attribute as italics rather than with underlining.

-ut This option indicates that *xterm* should not write a record into the the system *utmp* log file.

+ut This option indicates that *xterm* should write a record into the system *utmp* log file.

-vb This option indicates that a visual bell is preferred over an audible one. Instead of ringing the terminal bell whenever a Control-G is received, the window will be flashed.

+vb This option indicates that a visual bell should not be used.

-wc This option sets the **wideChars** resource. When **wideChars** is set, *xterm* maintains internal structures for 16-bit characters. If you do not set this resource to ''true'', *xterm* will ignore the escape sequence which turns UTF-8 mode on and off. The default is ''false''.

+wc This option resets the **wideChars** resource.

-wf This option 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 This option indicates that *xterm* should not wait before starting the subprocess.

-ziconbeep *percent*
Same as zIconBeep resource. If percent is non-zero, xterms that produce output while iconified will cause an XBell sound at the given volume and have "***" prepended to their icon titles. Most window managers will detect this change immediately, showing you which window has the output. (A similar feature was in x10 *xterm*.)

-C This option indicates that this 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.

-S*ccn* This option allows *xterm* to be used as an input and output channel for an existing program and is sometimes used in specialized applications. The option value specifies the last few letters of the name of a pseudo-terminal to use in slave mode, plus the number of the inherited file descriptor. If the option contains a ''/'' character, that delimits the characters used for the pseudo-terminal name from the file descriptor. Otherwise, exactly two characters are used from the option for the pseudo-terminal name, the remainder is the file descriptor. Examples:

```
-S123/45
-Sab34
```

Note that *xterm* does not close any file descriptor which it did not open for its own use. It is possible (though probably not portable) to have an application which passes an open file descriptor down to *xterm* past the initialization or the **-S** option to a process running in the *xterm*.

The following command line arguments are provided for compatibility with older versions. They may not be supported in the next release as the X Toolkit provides standard options that accomplish the same task.

%*geom* This option specifies the preferred size and position of the Tektronix window. It is shorthand for specifying the ‘‘**tekGeometry*’’ resource.

#*geom* This option specifies the preferred position of the icon window. It is shorthand for specifying the ‘‘**iconGeometry*’’ resource.

-T *string*
This option specifies the title for *xterm*’s windows. It is equivalent to **-title**.

-n *string* This option specifies the icon name for *xterm*’s windows. It is shorthand for specifying the ‘‘**iconName*’’ resource. Note that this is not the same as the toolkit option **-name** (see below). The default icon name is the application name.

-r This option indicates that reverse video should be simulated by swapping the foreground and background colors. It is equivalent to **-rv**.

-w *number*
This option specifies the width in pixels of the border surrounding the window. It is equivalent to **-borderwidth** or **-bw**.

The following standard X Toolkit command line arguments are commonly used with *xterm*:

-bd *color*
This option specifies the color to use for the border of the window. *xterm* uses the X Toolkit default, which is ‘‘XtDefaultForeground’’.

-bg *color*
This option specifies the color to use for the background of the window. The default is ‘‘XtDefaultBackground.’’

-bw *number*
This option specifies the width in pixels of the border surrounding the window.

This appears to be a legacy of older X releases. It sets the **borderWidth** resource of the shell widget, and may provide advice to your window manager to set the thickness of the window frame. Most window managers do not use this information. See the **-b** option, which controls the inner border of the *xterm* window.

-display *display*
This option specifies the X server to contact; see *X()*.

-fg *color*
This option specifies the color to use for displaying text. The default is ‘‘XtDefaultForeground.’’

-fn *font* This option specifies the font to be used for displaying normal text. The default is *fixed*.

-font *font*
This is the same as **-fn**.

-geometry *geometry*
This option specifies the preferred size and position of the VT102 window; see *X()*.

-iconic This option indicates that *xterm* should ask the window manager to start it as an icon rather than as the normal window.

-name *name*
This option specifies the application name under which resources are to be obtained, rather than the default executable file name. *Name* should not contain ‘‘.’’ or ‘‘*’’ characters.

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

+rv Disable the simulation of reverse video by swapping foreground and background colors.

-title *string*
This option 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.

-xrm *resourcestring*
This option specifies a resource string to be used. This is especially useful for setting resources that do not have separate command line options.

RESOURCES

The program understands all of the core X Toolkit resource names and classes. Application specific resources (e.g., "**XTerm.***NAME*") follow:

backarrowKeyIsErase (class **BackarrowKeyIsErase)**
Tie the VTxxx **backarrowKey** and **ptyInitialErase** resources together by setting the DECBKM state according to whether the initial value of stty erase is a backspace (8) or delete (127) character. The default is ‘‘false’’, which disables this feature.

hold (class **Hold)**
If true, *xterm* will not immediately destroy its window when the shell command completes. It will wait until you use the window manager to destroy/kill the window, or if you use the menu entries that send a signal, e.g., HUP or KILL. You may scroll back, select text, etc., to perform most graphical operations. Resizing the display will lose data, however, since this involves interaction with the shell which is no longer running.

hpFunctionKeys (class **HpFunctionKeys)**
Specifies whether or not HP Function Key escape codes should be generated for function keys instead of standard escape sequences. See also the **keyboardType** resource.

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.

keyboardType (class **KeyboardType)**
Enables one (or none) of the various keyboard-type resources: **hpFunctionKeys**, **scoFunctionKeys**, **sunFunctionKeys**, **tcapFunctionKeys** and **sunKeyboard**. The resource’s value should be one of the corresponding strings ‘‘hp’’, ‘‘sco’’, ‘‘sun’’, ‘‘tcap’’ or ‘‘vt220’’. The individual resources are provided for legacy support; this resource is simpler to use.

maxBufSize (class **MaxBufSize)**
Specify the maximum size of the input buffer. The default is 32768. You cannot set this to a value less than the **minBufSize** resource. It will be increased as needed to make that value evenly divide this one.

On some systems you may want to increase one or both of the **maxBufSize** and **minBufSize** resource values to achieve better performance if the operating system prefers larger buffer sizes.

messages (class **Messages)**
Specifies whether write access to the terminal is allowed initially. See **mesg**(1). The default is ‘‘true’’.

menuLocale (class **MenuLocale)**
Specify the locale used for character-set computations when loading the popup menus. Use this to improve initialization performance of the Athena popup menus, which may load unnecessary (and very large) fonts, e.g., in a locale having UTF-8 encoding. The default is an empty string, which uses the current locale setting.

Set it to ‘‘C’’ to achieve the best performance using the default menu resource settings. If you happen to be using localized menu resources, set the resource accordingly.

minBufSize (class **MinBufSize)**
Specify the minimum size of the input buffer, i.e., the amount of data that *xterm* requests on each read. The default is 4096. You cannot set this to a value less than 64.

ptyHandshake (class **PtyHandshake)**
If ‘‘true’’, *xterm* will perform handshaking during initialization to ensure that the parent and child processes update the **utmp** and **stty** state. See also **waitForMap** which waits for the pseudo-terminal’s notion of the screen size, and **ptySttySize** which resets the screen size after other terminal initialization is complete. The default is ‘‘true’’.

ptyInitialErase (class **PtyInitialErase)**
If ‘‘true’’, *xterm* will use the pseudo-terminal’s sense of the stty erase value. If ‘‘false’’, *xterm* will set the stty erase value to match its own configuration, using the **kb** string from the termcap entry as a reference, if available. In either case, the result is applied to the TERMCAP variable which *xterm* sets. See also the **ttyModes** resource, which may modify this. The default is ‘‘false’’.

ptySttySize (class **PtySttySize)**
If ‘‘true’’, *xterm* will reset the screen size after terminal initialization is complete. This is needed for some systems whose pseudo-terminals cannot propagate terminal characteristics. Where it is not needed, it can interfere with other methods for setting the intial screen size, e.g., via window manager interaction. See also **waitForMap** which waits for a handshake-message giving the pseudo-terminal’s notion of the screen size. The default is ‘‘false’’ on Linux and OS X systems, ‘‘true’’ otherwise.

sameName (class **SameName)**
If the value of this resource is ‘‘true’’, *xterm* does not send title and icon name change requests when the request would have no effect: the name is not changed. This has the advantage of preventing flicker and the disadvantage of requiring an extra round trip to the server to find out the previous value. In practice this should never be a problem. The default is ‘‘true’’.

scoFunctionKeys (class **ScoFunctionKeys)**
Specifies whether or not SCP Function Key escape codes should be generated for function keys instead of standard escape sequences. See also the **keyboardType** resource.

sessionMgt (class **SessionMgt)**
If the value of this resource is ‘‘true’’, *xterm* sets up session manager callbacks for **XtNdieCallback** and **XtNsaveCallback**. The default is ‘‘true’’.

sunFunctionKeys (class **SunFunctionKeys)**
Specifies whether or not Sun Function Key escape codes should be generated for function keys instead of standard escape sequences. See also the **keyboardType** resource.

sunKeyboard (class **SunKeyboard)**
Specifies whether or not Sun/PC keyboard layout should be assumed rather than DEC VT220. This causes the keypad ‘+’ to be mapped to ‘,’. and CTRL F1-F12 to F11-F20, depending on the setting of the **ctrlFKeys** resource. so *xterm* emulates a DEC VT220 more accurately. Otherwise (the default, with **sunKeyboard** set to ‘‘false’’), *xterm* uses PC-style bindings for the function keys and keypad.

PC-style bindings use the Shift, Alt, Control and Meta keys as modifiers for function-keys and keypad (see the document *Xterm Control Sequences* for details). The PC-style bindings are analogous to PCTerm, but not the same thing. Normally these bindings do not conflict with the use of the Meta key as described for the **eightBitInput** resource. If they do, note that the PC-style bindings are evaluated first. See also the **keyboardType** resource.

tcapFunctionKeys (class **TcapFunctionKeys)**
Specifies whether or not function key escape codes read from the termcap/terminfo entry should be generated for function keys instead of standard escape sequences. See also the **keyboardType** resource.

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 when displaying this application.

toolBar (class **ToolBar)**
Specifies whether or not the toolbar should be displayed. The default is ‘‘true.’’

ttyModes (class **TtyModes)**
Specifies a string containing terminal setting keywords and the characters to which they may be bound. Allowable keywords include: brk, dsusp, eof, eol, eol2, erase, erase2, flush, intr, kill, lnext, quit, rprnt, start, status, stop, susp, swtch and weras. Control characters may be specified as ^char (e.g., ^c or ^u) and **^?** may be used to indicate delete (127). Use **^-** to denote *undef*. Use **\034** to represent ^\, since a literal backslash in an X resource escapes the next character.

This is very useful for overriding the default terminal settings without having to do an *stty* every time an *xterm* is started. Note, however, that the stty program on a given host may use different keywords; *xterm*’s table is built-in.

If the **ttyModes** resource specifies a value for **erase**, that overrides the **ptyInitialErase** resource setting, i.e., *xterm* initializes the terminal to match that value.

useInsertMode (class **UseInsertMode)**
Force 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.’’

utmpDisplayId (class **UtmpDisplayId)**
Specifies whether or not *xterm* should try to record the display identifier (display number and screen number) as well as the hostname in the system *utmp* log file. The default is ‘‘true.’’

utmpInhibit (class **UtmpInhibit)**
Specifies whether or not *xterm* should try to record the user’s terminal in the system *utmp* log file. If true, *xterm* will not try. The default is ‘‘false.’’

waitForMap (class **WaitForMap)**
Specifies whether or not *xterm* should wait for the initial window map before starting the subprocess. This is part of the **ptyHandshake** logic. When *xterm* is directed to wait in this fashion, it passes the terminal size from the display end of the pseudo-terminal to the terminal I/O connection, e.g., according to the window manager. Otherwise, it uses the size as given in resource values or command-line option **-geom**. The default is ‘‘false.’’

zIconBeep (class **ZIconBeep)**
Same as -ziconbeep command line argument. If the value of this resource is non-zero, xterms that produce output while iconified will cause an XBell sound at the given volume and have "***" prepended to their icon titles. Most window managers will detect this change immediately, showing you which window has the output. (A similar feature was in x10 *xterm*.) The default is ‘‘false.’’

VT100 Widget Resources

The following resources are specified as part of the *vt100* widget (class *VT100*): These are specified by patterns such as "**XTerm.vt100.***NAME*":

activeIcon (class **ActiveIcon)**
Specifies whether or not active icon windows are to be used when the *xterm* window is iconified, if this feature is compiled into *xterm*. The active icon is a miniature representation of the content of the window and will update as the content changes. Not all window managers necessarily support application icon windows. Some window managers will allow you to enter keystrokes into the active icon window. The default is ‘‘false.’’

allowC1Printable (class **AllowC1Printable)**
If true, overrides the mapping of C1 controls (codes 128-159) to make them be treated as if they were printable characters. Although this corresponds to no particular standard, some users insist it is a VT100. The default is ‘‘false.’’

allowFontOps (class **AllowFontOps)**
Specifies whether control sequences that set/query the font should be allowed. The default is ‘‘false.’’

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. The default is ‘‘false.’’

allowTitleOps (class **AllowTitleOps)**
Specifies whether control sequences that modify the window title or icon name should be allowed. The default is ‘‘true.’’

allowWindowOps (class **AllowWindowOps)**
Specifies whether extended window control sequences (as used in dtterm) should be allowed. The default is ‘‘false.’’

altIsNotMeta (class **AltIsNotMeta**)
If ‘‘true’’, treat the Alt-key as if it were the Meta-key. Your keyboard may happen to be configured so they are the same. But if they are not, this allows you to use the same prefix- and shifting operations with the Alt-key as with the Meta-key. See **altSendsEscape** and **metaSendsEscape**. The default is ‘‘false.’’

altSendsEscape (class **AltSendsEscape**)
This is an additional keyboard operation that may be processed after the logic for **metaSendsEscape**. It is only available if the **altIsNotMeta** resource is set.

If ‘‘true’’, Alt characters (a character combined with the modifier associated with left/right Alt-keys) are converted into a two-character sequence with the character itself preceded by ESC. This applies as well to function key control sequences, unless *xterm* sees that **Alt** is used in your key translations. If ‘‘false’’, Alt characters input from the keyboard cause a shift to 8-bit characters (just like **metaSendsEscape**). By combining the Alt- and Meta-modifiers, you can create corresponding combinations of ESC-prefix and 8-bit characters. The default is ‘‘false.’’

alwaysBoldMode (class **AlwaysBoldMode)**
Specifies whether *xterm* should check if the normal and bold fonts are distinct before deciding whether to use overstriking to simulate bold fonts. If this resource is true, *xterm* does not make the check for distinct fonts when deciding how to handle the **boldMode** resource. The default is ‘‘false.’’

boldMode	*alwaysBoldMode*	*Comparison*	*Action*
false	false	ignored	use font
false	true	ignored	use font
true	false	same	overstrike
true	false	different	use font
true	true	ignored	overstrike

alwaysHighlight (class **AlwaysHighlight)**
Specifies whether or not *xterm* should always display a highlighted text cursor. By default (if this resource is false), a hollow text cursor is displayed whenever the pointer moves out of the window or the window loses the input focus. The default is ‘‘false.’’

alwaysUseMods (class **AlwaysUseMods)**
Override the **numLock** resource, telling *xterm* to use the Alt and Meta modifiers to construct parameters for function key sequences even if those modifiers appear in the translations resource. The default is ‘‘false.’’

answerbackString (class **AnswerbackString)**
Specifies the string that *xterm* sends in response to an ENQ (control/E) character from the host. The default is a blank string, i.e., ‘‘’’. A hardware VT100 implements this feature as a setup option.

appcursorDefault (class **AppcursorDefault)**
If ‘‘true,’’ the cursor keys are initially in application mode. This is the same as the VT102 private DECCKM mode, The default is ‘‘false.’’

appkeypadDefault (class **AppkeypadDefault)**
If ‘‘true,’’ the keypad keys are initially in application mode. The default is ‘‘false.’’

autoWrap (class **AutoWrap)**
Specifies whether or not auto-wraparound should be enabled. This is the same as the VT102 DECAWM. The default is ‘‘true.’’

awaitInput (class **AwaitInput)**
Specifies whether or not the *xterm* uses a 50 millisecond timeout to await input (i.e., to support the Xaw3d arrow scrollbar). The default is ‘‘false.’’

backarrowKey (class **BackarrowKey)**
Specifies whether the backarrow key transmits a backspace (8) or delete (127) character. This corresponds to the DECBKM control sequence. The default (backspace) is ‘‘true.’’ Pressing the control key toggles this behavior.

background (class **Background)**
Specifies the color to use for the background of the window. The default is ‘‘XtDefaultBackground.’’

bellIsUrgent (class **BellIsUrgent)**
Specifies whether to set the Urgency hint for the window manager when making a bell sound. The default is ‘‘false.’’

bellOnReset (class **BellOnReset)**
Specifies whether to sound a bell when doing a hard reset. The default is ‘‘true.’’

bellSuppressTime (class **BellSuppressTime)**
Number of milliseconds after a bell command is sent during which additional bells will be suppressed. Default is 200. If set non-zero, 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.

boldColors (class **ColorMode)**
Specifies whether to combine bold attribute with colors like the IBM PC, i.e., map colors 0 through 7 to colors 8 through 15. These normally are the brighter versions of the first 8 colors, hence bold. The default is ‘‘true.’’

boldFont (class **BoldFont)**
Specifies the name of the bold font to use instead of overstriking. There is no default for this resource.

boldMode (class **BoldMode)**
This specifies whether or not text with the bold attribute should be overstruck to simulate bold fonts if the resolved bold font is the same as the normal font. It may be desirable to disable bold fonts when color is being used for the bold attribute.

Note that *xterm* has one bold font which you may set explicitly. *Xterm* attempts to derive a bold font for the other font selections (**font1** through **font6**). If it cannot find a bold font, it will use the normal font. In each case (whether the explicit resource or the derived font), if the normal and bold fonts are distinct, this resource has no effect. The default is ‘‘true.’’

See the **alwaysBoldMode** resource which can modify the behavior of this resource.

Although *xterm* attempts to derive a bold font for other font selections, the font server may not cooperate. Since X11R6, bitmap fonts have been scaled. The font server claims to provide the bold font that *xterm* requests, but the result is not always readable. XFree86 provides a feature which can be used to suppress the scaling. In the X server's configuration file (e.g., "/etc/X11/XFree86"), you can add ":unscaled" to the end of the directory specification for the "misc" fonts, which comprise the fixed-pitch fonts that are used by *xterm*. For example

```
FontPath        "/usr/lib/X11/fonts/misc/"
```

would become

```
FontPath        "/usr/lib/X11/fonts/misc/:unscaled"
```

Depending on your configuration, the font server may have its own configuration file. The same ":unscaled" can be added to its configuration file at the end of the directory specification for "misc".

brokenLinuxOSC (class **BrokenLinuxOSC)**
If true, *xterm* applies a workaround to ignore malformed control sequences that a Linux script might send. Compare the palette control sequences documented in *console_codes* with ECMA-48. The default is ‘‘true.’’

brokenSelections (class **BrokenSelections)**
If true, *xterm* in 8-bit mode will interpret **STRING** selections as carrying text in the current locale's encoding. Normally **STRING** selections carry ISO-8859-1 encoded text. Setting this resource to ‘‘true’’ violates the ICCCM; it may, however, be useful for interacting with some broken X clients. The default is ‘‘false.’’

brokenStringTerm (class **BrokenStringTerm)**
provides a work-around for some ISDN routers which start an application control string without completing it. Set this to ‘‘true’’ if *xterm* appears to freeze when connecting. The default is ‘‘false.’’

c132 (class **C132)**
Specifies whether or not the VT102 DECCOLM escape sequence, used to switch between 80 and 132 columns, should be honored. The default is ‘‘false.’’

cacheDoublesize (class **CacheDoublesize)**
Tells whether to cache double-sized fonts by *xterm*. Set this to zero to disable doublesize fonts altogether.

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 the **CHARACTER CLASSES** section.

cjkWidth (class **CjkWidth)**
Specifies whether *xterm* should follow the traditional East Asian width convention. When turned on, characters with East Asian Ambiguous (A) category in UTR 11 have a column width of 2. You may have to set this option to ‘‘true’’ if you have some old East Asian terminal based programs that assume that line-drawing characters have a column width of 2. If this resource is false, the **mkWidth** resource controls the choice between the system's **wcwidth** and *xterm*'s built-in tables. The default is ‘‘false.’’

color0 (class **Color0)**

color1 (class **Color1)**

color2 (class **Color2)**

color3 (class **Color3)**

color4 (class **Color4)**

color5 (class **Color5)**

color6 (class **Color6)**

color7 (class **Color7)**
These specify the colors for the ISO-6429 extension. The defaults are, respectively, black, red3, green3, yellow3, a customizable dark blue, magenta3, cyan3, and gray90. The default shades of color are chosen to allow the colors 8-15 to be used as brighter versions.

color8 (class **Color8)**

color9 (class **Color9)**

color10 (class **Color10)**

color11 (class **Color11)**

color12 (class **Color12)**

color13 (class **Color13)**

color14 (class **Color14)**

color15 (class **Color15)**
These specify the colors for the ISO-6429 extension if the bold attribute is also enabled. The default resource values are respectively, gray30, red, green, yellow, a customizable light blue, magenta, cyan, and white.

color16 (class **Color16)**

through

color255 (class **Color255)**
These specify the colors for the 256-color extension. The default resource values are for colors 16 through 231 to make a 6x6x6 color cube, and colors 232 through 255 to make a grayscale ramp.

colorAttrMode (class **ColorAttrMode)**
Specifies whether **colorBD**, **colorBL**, **colorRV**, and **colorUL** should override ANSI colors. If not, these are displayed only when no ANSI colors have been set for the corresponding position. The default is ‘‘false.’’

colorBD (class **ColorBD)**
This specifies the color to use to display bold characters if the ‘‘colorBDMode’’ resource is enabled. The default is ‘‘XtDefaultForeground.’’

colorBDMode (class **ColorAttrMode)**
Specifies whether characters with the bold attribute should be displayed in color or as bold characters. Note that setting **colorMode** off disables all colors, including bold. The default is ‘‘false.’’

colorBL (class **ColorBL)**
This specifies the color to use to display blink characters if the ‘‘colorBLMode’’ resource is enabled. The default is ‘‘XtDefaultForeground.’’

colorBLMode (class **ColorAttrMode)**
Specifies whether characters with the blink attribute should be displayed in color. Note that setting **colorMode** off disables all colors, including this. The default is ‘‘false.’’

colorMode (class **ColorMode)**
Specifies whether or not recognition of ANSI (ISO-6429) color change escape sequences should be enabled. The default is ‘‘true.’’

colorRV (class **ColorRV)**
This specifies the color to use to display reverse characters if the ‘‘colorRVMode’’ resource is enabled. The default is ‘‘XtDefaultForeground.’’

colorRVMode (class **ColorAttrMode)**
Specifies whether characters with the reverse attribute should be displayed in color. Note that setting **colorMode** off disables all colors, including this. The default is ‘‘false.’’

colorUL (class **ColorUL)**
This specifies the color to use to display underlined characters if the ‘‘colorULMode’’ resource is enabled. The default is ‘‘XtDefaultForeground.’’

colorULMode (class **ColorAttrMode)**
Specifies whether characters with the underline attribute should be displayed in color or as underlined characters. Note that setting **colorMode** off disables all colors, including underlining. The default is ‘‘false.’’

combiningChars (class **CombiningChars)**
Specifies the number of wide-characters which can be stored in a cell to overstrike (combine) with the base character of the cell. This can be set to values in the range 0 to 4. The default is ‘‘2’’.

ctrlFKeys (class **CtrlFKeys)**
In VT220 keyboard mode (see **sunKeyboard** resource), specifies the amount by which to shift F1-F12 given a control modifier (CTRL). This allows you to generate key symbols for F10-F20 on a Sun/PC keyboard. The default is ‘‘10’’, which means that CTRL F1 generates the key symbol for F11.

curses (class **Curses)**
Specifies whether or not the last column bug in *more*(1) should be worked around. See the **-cu** option for details. The default is ‘‘false.’’

cursorBlink (class **CursorBlink)**
Specifies whether to make the cursor blink. The default is ‘‘false.’’

cursorColor (class **CursorColor)**
Specifies the color to use for the text cursor. The default is ‘‘XtDefaultForeground.’’ By default, *xterm* attempts to keep this color from being the same as the background color, since it draws the cursor by filling the background of a text cell. The same restriction applies to control sequences which may change this color.

Setting this resource overrides most of *xterm*’s adjustments to cursor color. It will still use reverse-video to disallow some cases, such as a black cursor on a black background.

cursorOffTime (class **CursorOffTime)**
Specifies the duration of the "off" part of the cursor blink cycle-time in milliseconds. The same timer is used for text blinking. The default is 300.

cursorOnTime (class **CursorOnTime)**
Specifies the duration of the "on" part of the cursor blink cycle-time, in milliseconds. The same timer is used for text blinking. The default is 600.

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.’’

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.’’

decTerminalID (class **DecTerminalID)**
Specifies the emulation level (100=VT100, 220=VT220, etc.), used to determine the type of response to a DA control sequence. Leading non-digit characters are ignored, e.g., "vt100" and "100" are the same. The default is 100.

deleteIsDEL (class **DeleteIsDEL)**
Specifies whether the Delete key on the editing keypad should send DEL (127) or the VT220-style Remove escape sequence. The default is ‘‘false,’’ for the latter.

dynamicColors (class **DynamicColors)**
Specifies whether or not escape sequences to change colors assigned to different attributes are recognized.

eightBitControl (class **EightBitControl**)
Specifies whether or not control sequences sent by the terminal should be eight-bit characters or escape sequences. The default is ‘‘false.’’

eightBitInput (class **EightBitInput**)
If ‘‘true’’, Meta characters (a single-byte character combined with the *Meta* modifier key) input from the keyboard are presented as a single character with the eighth bit turned on. The terminal is put into 8-bit mode. If ‘‘false’’, Meta characters are converted into a two-character sequence with the character itself preceded by ESC. On startup, *xterm* tries to put the terminal into 7-bit mode. The **metaSendsEscape** and **altSendsEscape** resources may override this. The default is ‘‘true.’’

Generally keyboards do not have a key labeled "Meta", but "Alt" keys are common, and they are conventionally used for "Meta". If they were synonymous, it would have been reasonable to name this resource "altSendsEscape", reversing its sense. For more background on this, see the **meta** function in curses.

Note that the *Alt* key is not necessarily the same as the *Meta* modifier. *xmodmap* lists your key modifiers. X defines modifiers for shift, (caps) lock and control, as well as 5 additional modifiers which are generally used to configure key modifiers. *xterm* inspects the same information to find the modifier associated with either *Meta* key (left or right), and uses that key as the *Meta* modifier. It also looks for the NumLock key, to recognize the modifier which is associated with that.

If your *xmodmap* configuration uses the same keycodes for Alt- and Meta-keys, *xterm* will only see the Alt-key definitions, since those are tested before Meta-keys. NumLock is tested first. It is important to keep these keys distinct; otherwise some of *xterm*’s functionality is not available.

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,’’ which means that they are accepted as is.

faceName (class **FaceName)**
Specify the pattern for fonts selected from the FreeType library if support for that library was compiled into *xterm*. There is no default. If not specified, or if there is no match for both normal and bold fonts, *xterm* uses the **font** and related resources.

faceNameDoublesize (class **FaceNameDoublesize)**
Specify an double-width font for cases where an application requires this, e.g., in CJK applications. There is no default. If the application uses double-wide characters and this resource is not given, *xterm* will use a scaled version of the font given by **faceName**.

faceSize (class **FaceSize)**
Specify the pointsize for fonts selected from the FreeType library if support for that library was compiled into *xterm*. The default is ‘‘14.’’ On the **VT Fonts** menu, this corresponds to the *Default* entry. You can specify the pointsize for TrueType fonts selected with the other size-related menu entries such as Medium, Huge, etc., by using one of the following resource values. If you do not specify a value, they default to ‘‘0.0’’, which causes *xterm* to use the ratio of font sizes from the bitmap font resources to obtain a TrueType pointsize.

faceSize1 (class **FaceSize1)**
Specifies the pointsize of the first alternative font.

faceSize2 (class **FaceSize2)**
Specifies the pointsize of the second alternative font.

faceSize3 (class **FaceSize3)**
Specifies the pointsize of the third alternative font.

faceSize4 (class **FaceSize4)**
Specifies the pointsize of the fourth alternative font.

faceSize5 (class **FaceSize5)**
Specifies the pointsize of the fifth alternative font.

faceSize6 (class **FaceSize6)**
Specifies the pointsize of the sixth alternative font.

font (class **Font)**
Specifies the name of the normal font. The default is ‘‘fixed.’’

See the discussion of the **locale** resource, which describes how this font may be overridden.

NOTE: some resource files use patterns such as
*font: fixed

which are overly broad, affecting both
xterm.vt100.font

and
xterm.vt100.utf8fonts.font

which is probably not what you intended.

font1 (class **Font1)**
Specifies the name of the first alternative font.

font2 (class **Font2)**
Specifies the name of the second alternative font.

font3 (class **Font3)**
Specifies the name of the third alternative font.

font4 (class **Font4)**
Specifies the name of the fourth alternative font.

font5 (class **Font5)**
Specifies the name of the fifth alternative font.

font6 (class **Font6)**
Specifies the name of the sixth alternative font.

fontDoublesize (class **FontDoublesize)**
Specifies whether *xterm* should attempt to use font scaling to draw doublesize characters. Some older font servers cannot do this properly, will return misleading font metrics. The default is ‘‘true’’. If disabled, *xterm* will simulate doublesize characters by drawing normal characters with spaces between them.

forceBoxChars (class **ForceBoxChars)**
Specifies whether *xterm* should assume the normal and bold fonts have VT100 line-drawing characters:

- The fixed-pitch ISO-8859-*-encoded fonts used by *xterm* normally have the VT100 line-drawing glyphs in cells 1-31. Other fixed-pitch fonts may be more attractive, but lack these glyphs.
- When using an ISO-10646-1 font and the **wideChars** resource is true, *xterm* uses the Unicode glyphs which match the VT100 line-drawing glyphs.

If ‘‘false’’, *xterm* checks for missing glyphs in the font and makes line-drawing characters directly as needed. If ‘‘true’’, *xterm* assumes the font does not contain the line-drawing characters, and draws them directly. The default is ‘‘false.’’

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 ‘‘XtDefaultForeground.’’

formatOtherKeys (class **FormatOtherKeys)**
Overrides the format of the escape sequence used to report modified keys with the *modifyOtherKeys* resource.

0 send modified keys as parameters for function-key 27 (default).

1 send modified keys as parameters for CSI u.

freeBoldBox (class **FreeBoldBox)**
Specifies whether *xterm* should assume the bounding boxes for normal and bold fonts are compatible. If ‘‘false’’, *xterm* compares them and will reject choices of bold fonts that do not match the size of the normal font. The default is ‘‘false’’, which means that the comparison is performed.

geometry (class **Geometry)**
Specifies the preferred size and position of the VT102 window. There is no default for this resource.

highlightColor (class **HighlightColor)**
Specifies the color to use for the background of selected (highlighted) text. If not specified (i.e., matching the default foreground), reverse video is used. The default is ‘‘XtDefaultForeground.’’

highlightColorMode (class **HighlightColorMode)**
Specifies whether *xterm* should use **highlightTextColor** and **highlightColor** to override the reversed foreground/background colors in a selection. The default is unspecified: at startup, *xterm* checks if those resources are set to something other than the default foreground and background colors. Setting this resource disables the check.

The following table shows the interaction of the highlighting resources, abbreviated as shown to fit in this page:

HCM
highlightColorMode

HR
highlightReverse

HBG
highlightColor

HFG
highlightTextColor

HCM	*HR*	*HBG*	*HFG*	*Highlight*
false	false	default	default	bg/fg
false	false	default	set	bg/fg
false	false	set	default	fg/HBG
false	false	set	set	fg/HBG
false	true	default	default	bg/fg
false	true	default	set	bg/fg
false	true	set	default	fg/HBG
false	true	set	set	fg/HBG

true	false	default	default	bg/fg
true	false	default	set	HFG/fg
true	false	set	default	bg/HBG
true	false	set	set	HFG/HBG
true	true	default	default	fg/fg (useless)
true	true	default	set	HFG/fg
true	true	set	default	fg/HBG
true	true	set	set	HFG/HBG
default	false	default	default	bg/fg
default	false	default	set	bg/fg
default	false	set	default	fg/HBG
default	false	set	set	HFG/HBG
default	true	default	default	bg/fg
default	true	default	set	bg/fg
default	true	set	default	fg/HBG
default	true	set	set	HFG/HBG

highlightReverse (class **HighlightReverse)**

Specifies whether *xterm* should reverse the selection foreground and background colors when selecting text with reverse-video attribute. This applies only to the **highlightColor** and **highlightTextColor** resources, e.g., to match the color scheme of *xwsh*. If ‘‘true’’, *xterm* reverses the colors, If ‘‘false’’, *xterm* does not reverse colors, The default is ‘‘true.’’

highlightSelection (class **HighlightSelection)**

If ‘‘false’’, selecting with the mouse highlights all positions on the screen between the beginning of the selection and the current position. If ‘‘true’’, *xterm* highlights only the positions that contain text that can be selected. The default is ‘‘false.’’

Depending on the way your applications write to the screen, there may be trailing blanks on a line. *Xterm* stores data as it is shown on the screen. Erasing the display changes the internal state of each cell so it is not considered a blank for the purpose of selection. Blanks written since the last erase are selectable. If you do not wish to have trailing blanks in a selection, use the **trimSelection** resource.

highlightTextColor (class **HighlightTextColor)**

Specifies the color to use for the foreground of selected (highlighted) text. If not specified (i.e., matching the default background), reverse video is used. The default is ‘‘XtDefaultBackground.’’

hpLowerleftBugCompat (class **HpLowerleftBugCompat)**

Specifies whether to work around a bug in HP’s *xdb*, which ignores termcap and always sends ESC F to move to the lower left corner. ‘‘true’’ causes *xterm* to interpret ESC F as a request to move to the lower left corner of the screen. The default is ‘‘false.’’

i18nSelections (class **I18nSelections)**

If false, *xterm* will never request the targets **COMPOUND_TEXT** or **TEXT**. The default is ‘‘true.’’ It may be set to false in order to work around ICCCM violations by other X clients.

iconBorderColor (class **BorderColor)**

Specifies the border color for the active icon window if this feature is compiled into *xterm*. Not all window managers will make the icon border visible.

iconBorderWidth (class **BorderWidth)**

Specifies the border width for the active icon window if this feature is compiled into *xterm*. The default is 2. Not all window managers will make the border visible.

iconFont (class **IconFont)**

Specifies the font for the miniature active icon window, if this feature is compiled into *xterm*. The default is "nil2".

initialFont (class **InitialFont)**
Specifies which of the VT100 fonts to use initially. Values are the same as for the *set-vt-font* action. The default is ‘‘d’’, i.e., "default".

internalBorder (class **BorderWidth)**
Specifies the number of pixels between the characters and the window border. The default is 2.

italicULMode (class **ColorAttrMode)**
Specifies whether characters with the underline attribute should be displayed in an italic font or as underlined characters. It is implemented only for TrueType fonts.

jumpScroll (class **JumpScroll)**
Specifies whether or not jump scroll should be used. This corresponds to the VT102 DECSCLM private mode. The default is ‘‘true.’’

keepSelection (class **KeepSelection)**
Specifies whether *xterm* will keep the selection even after the selected area was touched by some output to the terminal. The default is ‘‘false’’.

keyboardDialect (class **KeyboardDialect)**
Specifies the initial keyboard dialect, as well as the default value when the terminal is reset. The value given is the same as the final character in the control sequences which change character sets. The default is ‘‘B’’, which corresponds to US ASCII.

*name***Keymap (class** *Name***Keymap)**
See the discussion of the **keymap()** action.

limitResize (class **LimitResize)**
Limits resizing of the screen via control sequence to a given multiple of the display dimensions. The default is ‘‘1’’.

locale (class **Locale)**
Specifies how to use *luit*, an encoding converter between UTF-8 and locale encodings. The resource value (ignoring case) may be:

true
xterm will use the encoding specified by the users’ LC_CTYPE locale (i.e., LC_ALL, LC_CTYPE, or LANG variables) as far as possible. This is realized by always enabling UTF-8 mode and invoking *luit* in non-UTF-8 locales.

medium
xterm will follow users’ LC_CTYPE locale only for UTF-8, east Asian, and Thai locales, where the encodings were not supported by conventional 8bit mode with changing fonts. For other locales, *xterm* will use conventional 8bit mode.

checkfont
If mini-luit is compiled-in, *xterm* will check if a Unicode font has been specified. If so, it checks if the character encoding for the current locale is POSIX, Latin-1 or Latin-9, uses the appropriate mapping to support those with the Unicode font. For other encodings, *xterm* assumes that UTF-8 encoding is required.

false
xterm will use conventional 8bit mode or UTF-8 mode according to **utf8** resource or **-u8** option.

Any other value, e.g., ‘‘UTF-8’’ or ‘‘ISO8859-2’’, is assumed to be an encoding name; *luit* will be invoked to support the encoding. The actual list of supported encodings depends on *luit*. The default is ‘‘medium’’.

Regardless of your locale and encoding, you need an ISO-10646-1 font to display the result. Your configuration may not include this font, or locale-support by *xterm* may not be needed. At startup, *xterm* uses a mechanism equivalent to the **load-vt-fonts(utf8Fonts, Utf8Fonts)** action to load font name subresources of the VT100 widget. That is, resource patterns such as

"***vt100.utf8Fonts.font**" will be loaded, and (if this resource is enabled), override the normal fonts. If no subresources are found, the normal fonts such as "***vt100.font**", etc., are used. The resource files distributed with *xterm* use ISO-10646-1 fonts, but do not rely on them unless you are using the locale mechanism.

localeFilter (class **LocaleFilter)**
Specifies the file name for the encoding converter from/to locale encodings and UTF-8 which is used with the **-lc** option or **locale** resource. The help message shown by ''xterm -help'' lists the default value, which depends on your system configuration.

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.''

metaSendsEscape (class **MetaSendsEscape**)
If ''true'', Meta characters (a character combined with the *Meta* modifier key) are converted into a two-character sequence with the character itself preceded by ESC. This applies as well to function key control sequences, unless *xterm* sees that **Meta** is used in your key translations. If ''false'', Meta characters input from the keyboard are handled according to the **eightBitInput** resource. The default is ''false.''

mkSamplePass (class **MkSamplePass)**
If **mkSampleSize** is nonzero, and **mkWidth** (and **cjkWidth**) are false, on startup *xterm* compares its built-in tables to the system's wide character width data to decide if it will use the system's data. It tests the first **mkSampleSize** character values, and allows up to **mkSamplePass** mismatches before the test fails. The default (for the allowed number of mismatches) is 256.

mkSampleSize (class **MkSampleSize)**
With **mkSamplePass**, this specifies a startup test used for initializing wide character width calculations. The default (number of characters to check) is 1024.

mkWidth (class **MkWidth)**
Specifies whether *xterm* should use a built-in version of the wide character width calculation. See also the **cjkWidth** resource which can override this. The default is ''false.''

Here is a summary of the resources which control the choice of wide character width calculation:

cjkWidth	*mkWidth*	*Action*
false	false	use system tables subject to **mkSamplePass**
false	true	use built-in tables
true	false	use built-in CJK tables
true	true	use built-in CJK tables

modifyCursorKeys (class **ModifyCursorKeys**)
Tells how to handle the special case where Control-, Shift-, Alt- or Meta-modifiers are used to add a parameter to the escape sequence returned by a cursor-key. The default is ''2'':

Set it to -1 to disable it.
Set it to 0 to use the old/obsolete behavior.
Set it to 1 to prefix modified sequences with CSI.
Set it to 2 to force the modifier to be the second parameter if it would otherwise be the first.
Set it to 3 to mark the sequence with a '>' to hint that it is private.

modifyFunctionKeys (class **ModifyFunctionKeys**)
Tells how to handle the special case where Control-, Shift-, Alt- or Meta-modifiers are used to add a parameter to the escape sequence returned by a (numbered) function-key. The default is ''2''. The resource values are similar to **modifyCursorKeys**:

Set it to -1 to permit the user to use shift- and control-modifiers to construct function-key strings using the normal encoding scheme.
Set it to 0 to use the old/obsolete behavior.
Set it to 1 to prefix modified sequences with CSI.
Set it to 2 to force the modifier to be the second parameter if it would otherwise be the first.
Set it to 3 to mark the sequence with a '>' to hint that it is private.

If **modifyFunctionKeys** is zero, *xterm* uses Control- and Shift-modifiers to allow the user to construct numbered function-keys beyond the set provided by the keyboard:

Control
adds the value given by the **ctrlFKeys** resource.

Shift adds twice the value given by the **ctrlFKeys** resource.

Control/Shift
adds three times the value given by the **ctrlFKeys** resource.

As a special case, legacy (when **oldFunctionKeys** is true) or vt220 (when **sunKeyboard** is true) keyboards interpret only the Control-modifier when constructing numbered function-keys. This is done to provide compatible keyboards for DEC VT220 and related terminals that implement user-defined keys (UDK).

modifyOtherKeys (class **ModifyOtherKeys**)
Like **modifyCursorKeys**, tells *xterm* to construct an escape sequence for other keys (such as "2") when modified by Control-, Alt- or Meta-modifiers. This feature does not apply to function keys and well-defined keys such as ESC or the control keys. The default is ''0'':

0 disables this feature.

1 enables this feature for keys except for those with well-known behavior, e.g., Tab, Backarrow and some special control character cases, e.g., Control-Space to make a NUL.

2 enables this feature for keys including the exceptions listed.

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 by the **marginBell** resource. The default is 10.

numLock (class **NumLock)**
If ''true'', *xterm* checks if NumLock is used as a modifier (see *xmodmap*(1)). If so, this modifier is used to simplify the logic when implementing special NumLock for the **sunKeyboard** resource. Also (when **sunKeyboard** is false), similar logic is used to find the modifier associated with the left and right Alt keys. The default is ''true.''

oldXtermFKeys (class **OldXtermFKeys)**
If ''true'', *xterm* will use old-style control sequences for function keys F1 to F4, for compatibility with X Consortium *xterm*. Otherwise, it uses the VT100-style codes for PF1 to PF4. The default is ''false.''

on2Clicks (class **On2Clicks)**

on3Clicks (class **On3Clicks)**

on4Clicks (class **On4Clicks)**

on5Clicks (class **On5Clicks)**
Specify selection behavior in response to multiple mouse clicks. A single mouse click is always interpreted as described in the **SELECTION** section (see **POINTER USAGE**). Multiple mouse clicks (using the button which activates the **select-start** action) are interpreted according to the resource values of **on2Clicks**, etc. The resource value can be one of these:

word
Select a ''word'' as determined by the **charClass** resource. See the **CHARACTER CLASSES** section.

line
Select a line (counting wrapping).

group
Select a group of adjacent lines (counting wrapping). The selection stops on a blank line.

page
Select all visible lines, i.e., the page.

all Select all lines, i.e., including the saved lines.

regex
Select a ''word'' as determined by the regular expression which follows in the resource value.

none
No selection action is associated with this resource. *xterm* interprets it as the end of the list. For example, you may use it to disable triple (and higher) clicking by setting **on3Clicks** to ''none''.

The default values for **on2Clicks** and **on3Clicks** are ''word'' and ''line'', respectively. There is no default value for **on4Clicks** or **on5Clicks**, making those inactive. On startup, *xterm* determines the maximum number of clicks by the **on*X*Clicks** resource values which are set.

pointerColor (class **PointerColor)**
Specifies the foreground color of the pointer. The default is ''XtDefaultForeground.''

pointerColorBackground (class **PointerColorBackground)**
Specifies the background color of the pointer. The default is ''XtDefaultBackground.''

pointerMode (class **PointerMode)**
Specifies when the pointer may be hidden as the user types. It will be redisplayed if the user moves the mouse, or clicks one of its buttons.

0 never. This is the default.

1 the application running in *xterm* has not activated mouse mode.

2 always.

pointerShape (class **Cursor)**
Specifies the name of the shape of the pointer. The default is ''xterm.''

popOnBell (class **PopOnBell)**
Specifies whether the window would be raised when Control-G is received. The default is ''false.''

printAttributes (class **PrintAttributes)**
Specifies whether to print graphic attributes along with the text. A real DEC VTxxx terminal will print the underline, highlighting codes but your printer may not handle these. A ''0'' disables the attributes. A ''1'' prints the normal set of attributes (bold, underline, inverse and blink) as VT100-style control sequences. A ''2'' prints ANSI color attributes as well. The default is ''1.''

printerAutoClose (class **PrinterAutoClose)**
If ''true'', *xterm* will close the printer (a pipe) when the application switches the printer offline with a Media Copy command. The default is ''false.''

printerCommand (class **PrinterCommand)**
Specifies a shell command to which *xterm* will open a pipe when the first MC (Media Copy) command is initiated. The default is a blank string. If the resource value is given as a blank string, the printer is disabled.

printerControlMode (class **PrinterControlMode)**
Specifies the printer control mode. A ‘‘1’’ selects autoprint mode, which causes *xterm* to print a line from the screen when you move the cursor off that line with a line feed, form feed or vertical tab character, or an autowrap occurs. Autoprint mode is overridden by printer controller mode (a ‘‘2’’), which causes all of the output to be directed to the printer. The default is ‘‘0.’’

printerExtent (class **PrinterExtent)**
Controls whether a print page function will print the entire page (true), or only the the portion within the scrolling margins (false). The default is ‘‘false.’’

printerFormFeed (class **PrinterFormFeed)**
Controls whether a form feed is sent to the printer at the end of a print page function. The default is ‘‘false.’’

quietGrab (class **QuietGrab)**
Controls whether the cursor is repainted when *NotifyGrab* and *NotifyUngrab* event types are received during change of focus. The default is ‘‘false.’’

renderFont (class **RenderFont)**
If *xterm* is built with the Xft library, this controls whether the **faceName** resource is used. The default is ‘‘true.’’

resizeGravity (class **ResizeGravity)**
Affects the behavior when the window is resized to be taller or shorter. **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.

reverseVideo (class **ReverseVideo)**
Specifies whether or not reverse video should be simulated. The default is ‘‘false.’’

reverseWrap (class **ReverseWrap)**
Specifies whether or not reverse-wraparound should be enabled. This corresponds to *xterm*’s private mode 45. The default is ‘‘false.’’

rightScrollBar (class **RightScrollBar)**
Specifies whether or not the scrollbar should be displayed on the right rather than the left. 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.’’

scrollBarBorder (class **ScrollBarBorder)**
Specifies the width of the scrollbar border. Note that this is drawn to overlap the border of the *xterm* window. Modifying the scrollbar’s border affects only the line between the VT100 widget and the scrollbar. The default value is 1.

scrollKey (class **ScrollCond)**
Specifies whether or not pressing a key should automatically cause the scrollbar to go to the bottom of the scrolling region. This corresponds to *xterm*’s private mode 1011. 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.

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.''

selectToClipboard (class **SelectToClipboard)**
Tells *xterm* whether to use the PRIMARY or CLIPBOARD for SELECT tokens in the selection mechanism. The **set-select** action can change this at runtime, allowing the user to work with programs that handle only one of these mechanisms. The default is ''false'', which tells it to use PRIMARY.

shiftFonts (class **ShiftFonts)**
Specifies whether to enable the actions **larger-vt-font()** and **smaller-vt-font()**, which are normally bound to the shifted KP_Add and KP_Subtract. The default is ''true.''

showBlinkAsBold (class **ShowBlinkAsBold)**
Tells *xterm* whether to display text with blink-attribute the same as bold. If *xterm* has not been configured to support blinking text, the default is ''true.'', which corresponds to older versions of *xterm*, otherwise the default is ''false.''

showMissingGlyphs (class **ShowMissingGlyphs)**
Tells *xterm* whether to display a box outlining places where a character has been used that the font does not represent. The default is ''false.''

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. There is no default for this resource.

tekInhibit (class **TekInhibit)**
Specifies whether or not the escape sequence to enter Tektronix mode should be ignored. 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.''

tiXtraScroll (class **TiXtraScroll)**
Specifies whether *xterm* should scroll to a new page when processing the *ti* termcap entry, i.e., the private modes 47, 1047 or 1049. This is only in effect if **titeInhibit** is ''true'', because the intent of this option is to provide a picture of the full-screen application's display on the scrollback without wiping out the text that would be shown before the application was initialized. The default for this resource 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. *Xterm* supports terminfo in a different way, supporting composite control sequences (also known as private modes) 1047, 1048 and 1049 which have the same effect as the original 47 control sequence. The default

for this resource is ‘‘false.’’

translations (class **Translations)**
Specifies the key and button bindings for menus, selections, ‘‘programmed strings,’’ etc. The **translations** resource, which provides much of *xterm*’s configurability, is a feature of the X Toolkit Intrinsics library (Xt). See the **ACTIONS** section.

trimSelection (class **TrimSelection)**
If you set **highlightSelection**, you can see the text which is selected, including any trailing spaces. Clearing the screen (or a line) resets it to a state containing no spaces. Some lines may contain trailing spaces when an application writes them to the screen. However, you may not wish to paste lines with trailing spaces. If this resource is true, *xterm* will trim trailing spaces from text which is selected. It does not affect spaces which result in a wrapped line, nor will it trim the trailing newline from your selection. The default is ‘‘false.’’

underLine (class **UnderLine)**
This specifies whether or not text with the underline attribute should be underlined. It may be desirable to disable underlining when color is being used for the underline attribute. The default is ‘‘true.’’

useClipping (class **UseClipping)**
Tell *xterm* whether to use clipping to keep from producing dots outside the text drawing area. Originally used to work around for overstriking effects, this is also needed to work with some incorrectly-sized fonts. The default is ‘‘true.’’

utf8 (class **Utf8)**
This specifies whether *xterm* will run in UTF-8 mode. If you set this resource, *xterm* also sets the **wideChars** resource as a side-effect. The resource is an integer, expected to range from 0 to 3:

0 UTF-8 mode is initially off. The command-line option **+u8** sets the resource to this value. Escape sequences for turning UTF-8 mode on/off are allowed.

1 UTF-8 mode is initially on. Escape sequences for turning UTF-8 mode on/off are allowed.

2 The command-line option **-u8** sets the resource to this value. Escape sequences for turning UTF-8 mode on/off are ignored.

3 This is the default value of the resource. It is changed during initialization depending on whether the **locale** resource was set, to 0 or 2. See the **locale** resource for additional discussion of non-UTF-8 locales.

If you want to set the value of **utf8**, it should be in this range. Other nonzero values are treated the same as ‘‘1’’, i.e., UTF-8 mode is initially on, and escape sequences for turning UTF-8 mode on/off are allowed.

utf8Fonts (class **Utf8Fonts)**
See the discussion of the **locale** resource.

utf8Latin1 (class **Utf8Latin1)**
If true, allow an ISO-8859-1 *normal* font to be combined with an ISO-10646 font if the latter is given via the **-fw** option or its corresponding resource value. The default is ‘‘false.’’

utf8Title (class **Utf8Title)**
Applications can set *xterm*’s title by writing a control sequence. Normally this control sequence follows the VT220 convention, which encodes the string in ISO-8859-1 and allows for an 8-bit string terminator. If *xterm* is started in a UTF-8 locale, it translates the ISO-8859-1 string to UTF-8 to work with the X libraries which assume the string is UTF-8.

However, some users may wish to write a title string encoded in UTF-8. Set this resource to ‘‘true’’ to allow UTF-8 encoded title strings. That cancels the translation to UTF-8, allowing UTF-8 strings to be displayed as is.

The default is ‘‘false.’’

veryBoldColors (class **VeryBoldColors)**
Specifies whether to combine video attributes with colors specified by **colorBD**, **colorBL**, **colorRV** and **colorUL**. The resource value is the sum of values for each attribute:
1 for reverse,
2 for underline,
4 for bold and
8 for blink.

The default is ‘‘0.’’

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.’’

visualBellDelay (class **VisualBellDelay)**
Number of milliseconds to delay when displaying a visual bell. Default is 100. If set to zero, no visual bell is displayed. This is useful for very slow displays, e.g., an LCD display on a laptop.

vt100Graphics (class **VT100Graphics)**
This specifies whether *xterm* will interpret VT100 graphic character escape sequences while in UTF-8 mode. The default is ‘‘true’’, to provide support for various legacy applications.

wideBoldFont (class **WideBoldFont)**
This option specifies the font to be used for displaying bold wide text. By default, it will attempt to use a font twice as wide as the font that will be used to draw bold text. If no doublewidth font is found, it will improvise, by stretching the bold font.

wideChars (class **WideChars)**
Specifies if *xterm* should respond to control sequences that process 16-bit characters. The default is ‘‘false.’’

wideFont (class **WideFont)**
This option specifies the font to be used for displaying wide text. By default, it will attempt to use a font twice as wide as the font that will be used to draw normal text. If no doublewidth font is found, it will improvise, by stretching the normal font.

ximFont (class **XimFont)**
This option specifies the font to be used for displaying the preedit string in the "OverTheSpot" input method.

In "OverTheSpot" preedit type, the preedit (preconversion) string is displayed at the position of the cursor. It is the XIM server’s responsibility to display the preedit string. The XIM client must inform the XIM server of the cursor position. For best results, the preedit string must be displayed with a proper font. Therefore, *xterm* informs the XIM server of the proper font. The font is be supplied by a "fontset", whose default value is "*". This matches every font, the X library automatically chooses fonts with proper charsets. The **ximFont** resource is provided to override this default font setting.

Tek4014 Widget Resources
The following resources are specified as part of the *tek4014* widget (class *Tek4014*). These are specified by patterns such as "**XTerm.tek4014.***NAME*":

font2 (class **Font)**
Specifies font number 2 to use in the Tektronix window.

font3 (class **Font)**
Specifies font number 3 to use in the Tektronix window.

fontLarge (class **Font)**
Specifies the large font to use in the Tektronix window.

fontSmall (class **Font)**
Specifies the small font to use in the Tektronix window.

ginTerminator (class **GinTerminator)**
Specifies what character(s) should follow a GIN report or status report. The possibilities are ‘‘none,’’ which sends no terminating characters, ‘‘CRonly,’’ which sends CR, and ‘‘CR&EOT,’’ which sends both CR and EOT. The default is ‘‘none.’’

height (class **Height)**
Specifies the height of the Tektronix window in pixels.

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.’’

width (class **Width)**
Specifies the width of the Tektronix window in pixels.

Menu Resources

The resources that may 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. Resources named "**line***N*" where *N* is a number are separators with class **SmeLine**.

The *mainMenu* has the following entries:

toolbar (class **SmeBSB)**
This entry invokes the **set-toolbar(toggle)** action.

securekbd (class **SmeBSB)**
This entry invokes the **secure()** action.

allowsends (class **SmeBSB)**
This entry invokes the **allow-send-events(toggle)** action.

redraw (class **SmeBSB)**
This entry invokes the **redraw()** action.

logging (class **SmeBSB)**
This entry invokes the **logging(toggle)** action.

print (class **SmeBSB)**
This entry invokes the **print()** action.

print-redir (class **SmeBSB)**
This entry invokes the **print-redir()** action.

8-bit-control (class **SmeBSB)**
This entry invokes the **set-8-bit-control(toggle)** action.

backarrow key (class **SmeBSB)**
This entry invokes the **set-backarrow(toggle)** action.

num-lock (class **SmeBSB)**
This entry invokes the **set-num-lock(toggle)** action.

alt-esc (class **SmeBSB)**
This entry invokes the **alt-sends-escape(toggle)** action.

meta-esc (class **SmeBSB)**
This entry invokes the **meta-sends-escape(toggle)** action.

delete-is-del (class **SmeBSB)**
This entry invokes the **delete-is-del(toggle)** action.

oldFunctionKeys (class **SmeBSB)**
This entry invokes the **old-function-keys(toggle)** action.

hpFunctionKeys (class **SmeBSB)**
This entry invokes the **hp-function-keys(toggle)** action.

scoFunctionKeys (class **SmeBSB)**
This entry invokes the **sco-function-keys(toggle)** action.

sunFunctionKeys (class **SmeBSB)**
This entry invokes the **sun-function-keys(toggle)** action.

sunKeyboard (class **SmeBSB)**
This entry invokes the **sunKeyboard(toggle)** action.

suspend (class **SmeBSB)**
This entry invokes the **send-signal(tstp)** action on systems that support job control.

continue (class **SmeBSB)**
This entry invokes the **send-signal(cont)** action on systems that support job control.

interrupt (class **SmeBSB)**
This entry invokes the **send-signal(int)** action.

hangup (class **SmeBSB)**
This entry invokes the **send-signal(hup)** action.

terminate (class **SmeBSB)**
This entry invokes the **send-signal(term)** action.

kill (class **SmeBSB)**
This entry invokes the **send-signal(kill)** action.

quit (class **SmeBSB)**
This entry invokes the **quit()** action.

The *vtMenu* has the following entries:

scrollbar (class **SmeBSB)**
This entry invokes the **set-scrollbar(toggle)** action.

jumpscroll (class **SmeBSB)**
This entry invokes the **set-jumpscroll(toggle)** action.

reversevideo (class **SmeBSB)**
This entry invokes the **set-reverse-video(toggle)** action.

autowrap (class **SmeBSB)**
This entry invokes the **set-autowrap(toggle)** action.

reversewrap (class **SmeBSB)**
This entry invokes the **set-reversewrap(toggle)** action.

autolinefeed (class **SmeBSB)**
This entry invokes the **set-autolinefeed(toggle)** action.

appcursor (class **SmeBSB)**
This entry invokes the **set-appcursor(toggle)** action.

appkeypad (class **SmeBSB)**
This entry invokes the **set-appkeypad(toggle)** action.

scrollkey (class **SmeBSB)**
This entry invokes the **set-scroll-on-key(toggle)** action.

scrollttyoutput (class **SmeBSB)**
This entry invokes the **set-scroll-on-tty-output(toggle)** action.

allow132 (class **SmeBSB)**
This entry invokes the **set-allow132(toggle)** action.

cursesemul (class **SmeBSB)**
This entry invokes the **set-cursesemul(toggle)** action.

visualbell (class **SmeBSB)**
This entry invokes the **set-visualbell(toggle)** action.

bellIsUrgent (class **SmeBSB)**
This entry invokes the **set-bellIsUrgent(toggle)** action.

poponbell (class **SmeBSB)**
This entry invokes the **set-poponbell(toggle)** action.

cursorblink (class **SmeBSB)**
This entry invokes the **set-cursorblink(toggle)** action.

titeInhibit (class **SmeBSB)**
This entry invokes the **set-titeInhibit(toggle)** action.

activeicon (class **SmeBSB)**
This entry toggles active icons on and off if this feature was compiled into *xterm*. It is enabled only if *xterm* was started with the command line option +ai or the **activeIcon** resource is set to ‘‘true.’’

softreset (class **SmeBSB)**
This entry invokes the **soft-reset()** action.

hardreset (class **SmeBSB)**
This entry invokes the **hard-reset()** action.

clearsavedlines (class **SmeBSB)**
This entry invokes the **clear-saved-lines()** action.

tekshow (class **SmeBSB)**
This entry invokes the **set-visibility(tek,toggle)** action.

tekmode (class **SmeBSB)**
This entry invokes the **set-terminal-type(tek)** action.

vthide (class **SmeBSB)**
This entry invokes the **set-visibility(vt,off)** action.

altscreen (class **SmeBSB)**
This entry invokes the **set-altscreen(toggle)** action.

The *fontMenu* has the following entries:

fontdefault (class **SmeBSB)**
This entry invokes the **set-vt-font(d)** action.

font1 (class **SmeBSB)**
This entry invokes the **set-vt-font(1)** action.

font2 (class **SmeBSB)**
This entry invokes the **set-vt-font(2)** action.

font3 (class **SmeBSB)**
This entry invokes the **set-vt-font(3)** action.

font4 (class **SmeBSB)**
This entry invokes the **set-vt-font(4)** action.

font5 (class **SmeBSB)**
This entry invokes the **set-vt-font(5)** action.

font6 (class **SmeBSB)**
This entry invokes the **set-vt-font(6)** action.

fontescape (class **SmeBSB)**
This entry invokes the **set-vt-font(e)** action.

fontsel (class **SmeBSB)**
This entry invokes the **set-vt-font(s)** action.

font-linedrawing (class **SmeBSB)**
This entry invokes the **set-font-linedrawing(s)** action.

font-doublesize (class **SmeBSB)**
This entry invokes the **set-font-doublesize(s)** action.

render-font (class **SmeBSB)**
This entry invokes the **set-render-font(s)** action.

utf8-mode (class **SmeBSB)**
This entry invokes the **set-utf8-mode(s)** action.

utf8-title (class **SmeBSB)**
This entry invokes the **set-utf8-title(s)** action.

The *tekMenu* has the following entries:

tektextlarge (class **SmeBSB)**
This entry invokes the **set-tek-text(large)** action.

tektext2 (class **SmeBSB)**
This entry invokes the **set-tek-text(2)** action.

tektext3 (class **SmeBSB)**
This entry invokes the **set-tek-text(3)** action.

tektextsmall (class **SmeBSB)**
This entry invokes the **set-tek-text(small)** action.

tekpage (class **SmeBSB)**
This entry invokes the **tek-page()** action.

tekreset (class **SmeBSB)**
This entry invokes the **tek-reset()** action.

tekcopy (class **SmeBSB)**
This entry invokes the **tek-copy()** action.

vtshow (class **SmeBSB)**
This entry invokes the **set-visibility(vt,toggle)** action.

vtmode (class **SmeBSB)**
This entry invokes the **set-terminal-type(vt)** action.

tekhide (class **SmeBSB)**
This entry invokes the **set-visibility(tek,toggle)** action.

Scrollbar Resources
The following resources are useful when specified for the Athena Scrollbar widget:

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.

POINTER USAGE

Once the VT102 window is created, *xterm* allows you to select text and copy it within the same or other windows.

SELECTION

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 one (usually left) is used to save text into the cut buffer. Move the cursor to beginning of the text, and then hold the button down while moving the cursor to the end of the region and releasing 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. Normally (but see the discussion of **on2Clicks**, etc):

- 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. Logical words and lines selected by double- or triple-clicking may wrap across more than one screen line if lines were wrapped by *xterm* itself rather than by the application running in the window. 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 two (usually middle) ‘types’ (pastes) the text from the PRIMARY selection, if any, otherwise from the cut buffer, inserting it as keyboard input.

Pointer button three (usually right) extends the current selection. (Without loss of generality, 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 that the last selection or extension was performed in; 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 cut buffers are globally shared among different applications, you may regard each 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.

SCROLLING

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 one with the pointer in the scroll region moves the adjacent line to the top of the display window.

Clicking button three moves the top line of the display window down to the pointer position.

Clicking button two moves the display to a position in the saved text that corresponds to the pointer’s position in the scrollbar.

TEKTRONIX POINTER

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 one, two, or three will return the letters ‘l’, ‘m’, and ‘r’, respectively. If the ‘shift’ key is pressed when a pointer button is pressed, the corresponding upper case letter is sent. To distinguish a pointer button from a key, the high bit of the character is set (but this is bit is normally stripped unless the terminal mode is RAW; see *tty*(4) for details).

MENUS

Xterm has four menus, named *mainMenu*, *vtMenu*, *fontMenu*, and *tekMenu*. Each menu pops up under the correct combinations of key and button presses. Each menu is divided into sections, separated by a horizontal line. Some menu entries correspond to modes that can be altered. A check mark appears next to a mode that is currently active. Selecting one of these modes toggles its state. Other menu entries are commands; selecting one of these performs the indicated function.

All of the menu entries correspond to X actions. In the list below, the menu label is shown followed by the action's name in parenthesis.

Main Options

The *xterm mainMenu* pops up when the ''control'' key and pointer button one are pressed in a window. This menu contains items that apply to both the VT102 and Tektronix windows. There are several sections:

Commands for managing X events:

Toolbar Clicking on the "Toolbar" menu entry hides the toolbar if it is visible, and shows it if it is not.

Secure Keyboard (securekbd)
The **Secure Keyboard** mode is helpful when typing in passwords or other sensitive data in an unsecure environment; see **SECURITY** below (but read the limitations carefully).

Allow SendEvents (allowsends)
Specifies whether or not synthetic key and button events generated using the X protocol SendEvent request should be interpreted or discarded. This corresponds to the **allowSendEvents** resource.

Redraw Window (redraw)
Forces the X display to repaint; useful in some environments.

Commands for capturing output:

Log to File (logging)
Captures text sent to the screen in a logfile, as in the **-l** logging option.

Print Window (print)
Sends the text of the current window to the program given in the **printerCommand** resource.

Redirect to Printer (print-redir)
This sets the **printerControlMode** to 0 or 2. You can use this to turn the printer on as if an application had sent the appropriate control sequence. It is also useful for switching the printer off if an application turns it on without resetting the print control mode.

Modes for setting keyboard style:

8-Bit Controls (8-bit-control)
Enabled for VT220 emulation, this controls whether *xterm* will send 8-bit control sequences rather than using 7-bit (ASCII) controls, e.g., sending a byte in the range 128-159 rather than the escape character followed by a second byte. *Xterm* always interprets both 8-bit and 7-bit control sequences (see the document *Xterm Control Sequences*). This corresponds to the **eightBitControl** resource.

Backarrow Key (BS/DEL) (backarrow key)
Modifies the behavior of the backarrow key, making it transmit either a backspace (8) or delete (127) character. This corresponds to the **backarrowKey** resource.

Alt/NumLock Modifiers (num-lock)
Controls the treatment of Alt- and NumLock-key modifiers. This corresponds to the **numLock** resource.

Meta Sends Escape (meta-esc)
Controls whether *Meta* keys are converted into a two-character sequence with the character itself preceded by ESC. This corresponds to the **metaSendsEscape** resource.

Delete is DEL (delete-is-del)
Controls whether the Delete key on the editing keypad should send DEL (127) or the VT220-style Remove escape sequence. This corresponds to the **deleteIsDEL** resource.

Old Function-Keys (oldFunctionKeys)

HP Function-Keys (hpFunctionKeys)

SCO Function-Keys (scoFunctionKeys)

Sun Function-Keys (sunFunctionKeys)

VT220 Keyboard (sunKeyboard)
These act as a radio-button, selecting one style for the keyboard layout. It corresponds to more than one resource setting: **sunKeyboard**, **sunFunctionKeys**, **scoFunctionKeys** and **hpFunctionKeys** ."

Commands for process signalling:

Send STOP Signal (suspend)

Send CONT Signal (continue)

Send INT Signal (interrupt)

Send HUP Signal (hangup)

Send TERM Signal (terminate)

Send KILL Signal (kill)
These send the SIGTSTP, SIGCONT, SIGINT, SIGHUP, SIGTERM and SIGKILL signals respectively, to the process group of the process running under *xterm* (usually the shell). The **SIGCONT** function is especially useful if the user has accidentally typed CTRL-Z, suspending the process.

Quit (quit)
Stop processing X events except to support the **-hold** option, and then send a SIGHUP signal to the the process group of the process running under *xterm* (usually the shell).

VT Options

The *vtMenu* sets various modes in the VT102 emulation, and is popped up when the ‘‘control’’ key and pointer button two are pressed in the VT102 window.

VT102/VT220 Modes:

Enable Scrollbar (scrollbar)
Enable (or disable) the scrollbar. This corresponds to the **-sb** option and the **scrollBar** resource.

Enable Jump Scroll (jumpscroll)
Enable (or disable) jump scrolling. This corresponds to the **-j** option and the **jumpScroll** resource.

Enable Reverse Video (reversevideo)
Enable (or disable) reverse-video. This corresponds to the **-rv** option and the **reverseVideo** resource.

Enable Auto Wraparound (autowrap)
Enable (or disable) auto-wraparound. This corresponds to the **-aw** option and the **autoWrap** resource.

Enable Reverse Wraparound (reversewrap)
: Enable (or disable) reverse wraparound. This corresponds to the **-rw** option and the **reverseWrap** resource.

Enable Auto Linefeed (autolinefeed)
: Enable (or disable) auto-linefeed. This is the VT102 NEL function, which causes the emulator to emit a linefeed after each carriage return. There is no corresponding command-line option or resource setting.

Enable Application Cursor Keys (appcursor)
: Enable (or disable) application cursor keys. This corresponds to the **appcursorDefault** resource. There is no corresponding command-line option.

Enable Application Keypad (appkeypad)
: Enable (or disable) application keypad keys. This corresponds to the **appkeypadDefault** resource. There is no corresponding command-line option.

Scroll to Bottom on Key Press (scrollkey)
: Enable (or disable) scrolling to the bottom of the scrolling region on a keypress. This corresponds to the **-sk** option and the **scrollKey** resource.

Scroll to Bottom on Tty Output (scrollttyoutput)
: Enable (or disable) scrolling to the bottom of the scrolling region on output to the terminal. This corresponds to the **-si** option and the **scrollTtyOutput** resource.

Allow 80/132 Column Switching (allow132)
: Enable (or disable) switching between 80 and 132 columns. This corresponds to the **-132** option and the **c132** resource.

Keep Selection (keepSelection)
: Tell *xterm* whether to disown the selection when it stops highlighting it, e.g., when an application modifies the display so that it no longer matches the text which has been highlighted. As long as *xterm* continues to own the selection, it can provide the corresponding text to other clients via cut/paste. This corresponds to the **keepSelection** resource. There is no corresponding command-line option.

Select to Clipboard (selectToClipboard)
: Tell *xterm* whether to use the PRIMARY or CLIPBOARD for SELECT tokens in the **translations** resource which maps keyboard and mouse actions to select/paste actions. This corresponds to the **selectToClipboard** resource. There is no corresponding command-line option.

Enable Visual Bell (visualbell)
: Enable (or disable) visible bell (i.e., flashing) instead of an audible bell. This corresponds to the **-vb** option and the **visualBell** resource.

Enable Bell Urgency (bellIsUrgent)
: Enable (or disable) Urgency window manager hint when Control-G is received. This corresponds to the **bellIsUrgent** resource.

Enable Pop on Bell (poponbell)
: Enable (or disable) raising of the window when Control-G is received. This corresponds to the **-pop** option and the **popOnBell** resource.

Enable Blinking Cursor (cursorblink)
: Enable (or disable) the blinking-cursor feature. This corresponds to the **-bc** option and the **cursorBlink** resource. There is also an escape sequence (see the document *Xterm Control Sequences*). The menu entry and the escape sequence states are XOR'd: if both are enabled, the cursor will not blink, if only one is enabled, the cursor will blink.

Enable Alternate Screen Switching (titeInhibit)
Enable (or disable) switching between the normal and alternate screens. This corresponds to the **titeInhibit** resource. There is no corresponding command-line option.

Enable Active Icon (activeicon)
Enable (or disable) the active-icon feature. This corresponds to the **-ai** option and the **activeIcon** resource.

VT102/VT220 Commands:

Do Soft Reset (softreset)
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). This corresponds to the VT220 DECSTR control sequence.

Do Full Reset (hardreset)
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 finished processing the command line options. This corresponds to the VT102 RIS control sequence, with a few obvious differences. For example, your session is not disconnected as a real VT102 would do.

Reset and Clear Saved Lines (clearsavedlines)
Perform a full reset, and also clear the saved lines.

Commands for setting the current screen:

Show Tek Window (tekshow)
When enabled, pops the Tektronix 4014 window up (makes it visible). When disabled, hides the Tektronix 4014 window.

Switch to Tek Mode (tekmode)
When enabled, pops the Tektronix 4014 window up if it is not already visible, and switches the input stream to that window. When disabled, hides the Tektronix 4014 window and switches input back to the VTxxx window.

Hide VT Window (vthide)
When enabled, hides the VTxxx window, shows the Tektronix 4014 window if it was not already visible and switches the input stream to that window. When disabled, shows the VTxxx window, and switches the input stream to that window.

Show Alternate Screen (altscreen)
When enabled, shows the alternate screen. When disabled, shows the normal screen. Note that the normal screen may have saved lines; the alternate screen does not.

VT Fonts

The *fontMenu* pops up when when the ‘‘control’’ key and pointer button three are pressed in a window. It sets the font used in the VT102 window, or modifies the way the font is specified or displayed. There are three sections.

The first section allows you to select the font from a set of alternatives:

Default (fontdefault)
Set the font to the default, i.e., that given by the ***VT100.font** resource.

Unreadable (font1)
Set the font to that given by the ***VT100.font1** resource.

Tiny (font2)
Set the font to that given by the ***VT100.font2** resource.

Small (font3)
Set the font to that given by the ***VT100.font3** resource.

Medium (font4)
Set the font to that given by the ***VT100.font4** resource.

Large (font5)
Set the font to that given by the ***VT100.font5** resource.

Huge (font6)
Set the font to that given by the ***VT100.font6** resource.

Escape Sequence
This allows you to set the font last specified by the Set Font escape sequence (see the document *Xterm Control Sequences*).

Selection (fontsel)
This allows you to set the font specified the current selection as a font name (if the PRIMARY selection is owned).

The second section allows you to modify the way it is displayed:

Line-Drawing Characters (font-linedrawing)
When set, tells *xterm* to draw its own line-drawing characters. Otherwise it relies on the font containing these. Compare to the **forceBoxChars** resource.

Doublesized Characters (font-doublesize)
When set, *xterm* may ask the font server to produce scaled versions of the normal font, for VT102 double-size characters.

The third section allows you to modify the way it is specified:

TrueType Fonts (render-font)
If the **renderFont** and corresponding resources were set, this is a further control whether *xterm* will actually use the Xft library calls to obtain a font.

UTF-8 (utf8-mode)
This controls whether *xterm* uses UTF-8 encoding of input/output. It is useful for temporarily switching *xterm* to display text from an application which does not follow the locale settings.

TEK Options

The *tekMenu* sets various modes in the Tektronix emulation, and is popped up when the ‘‘control’’ key and pointer button two are pressed in the Tektronix window. The current font size is checked in the modes section of the menu.

Large Characters (tektextlarge)

#2 Size Characters (tektext2)

#3 Size Characters (tektext3)

Small Characters (tektextsmall)

Commands:

PAGE (tekpage)
Clear the Tektronix window.

RESET (tekreset)

COPY (tekcopy)

Windows:

Show VT Window (vtshow)

Switch to VT Mode (vtmode)

Hide Tek Window (tekhide)

SECURITY

X environments differ in their security consciousness. Most 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 only using a host-based mechanism to control access to the server (see *xhost(1)*), then if you enable access for a host and other users are also permitted to run clients on that same host, it is possible that someone can run an application which uses the basic services of the X protocol to snoop on your activities, potentially capturing a transcript of everything you type at the keyboard. Any process which has access to your X display can manipulate it in ways that you might not anticipate, even redirecting your keyboard to itself and sending events to your application’s windows. This is true even with the ‘‘magic cookie’’ authorization scheme. While the **allowSendEvents** provides some protection against rogue applications tampering with your programs, guarding against a snooper is harder.

The possibility of an application spying on your keystrokes 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 is provided by X. Given all of these caveats, a simple mechanism exists for protecting keyboard input in *xterm*.

The *xterm* menu (see **MENUS** above) contains a **Secure Keyboard** entry which, when enabled, attempts to ensure 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. This ensures that you know which window is accepting your keystrokes. It cannot ensure that there are no processes which have access to your X display that might be observing the keystrokes as well.

Only one X client at a time can grab 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 **Reverse Video** entry in the **Modes** 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 title bar 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 left pointer button twice in rapid succession (double-clicking) causes all characters of the same class (e.g., letters, white space, punctuation) to be selected as a ‘‘word’’. 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 a series of comma-separated of *range*:*value* pairs. The *range* is either a single number or *low-high* in the range of 0 to 65535, corresponding to the 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. When not in UTF-8 mode, only the first 256 bytes of this table will be used.

The default table starts as follows - static int charClass[256] = { /* 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, /* x80 x81 x82 x83 IND NEL SSA ESA */
  1,   1,   1,   1,   1,   1,   1,   1, /* HTS HTJ VTS PLD PLU  RI SS2 SS3 */
  1,   1,   1,   1,   1,   1,   1,   1, /* DCS PU1 PU2 STS CCH  MW SPA EPA */
  1,   1,   1,   1,   1,   1,   1,   1, /* x98 x99 x9A CSI  ST OSC  PM APC */
  1,   1,   1,   1,   1,   1,   1,   1, /*  -   i  c/   L  ox  Y-   |  So */
160, 161, 162, 163, 164, 165, 166, 167, /*  ..  c0  ip  <<   _       R0   - */
168, 169, 170, 171, 172, 173, 174, 175, /*  o  +-   2   3   '   u  q|   . */
176, 177, 178, 179, 180, 181, 182, 183, /*  ,   1   2  >>  1/4 1/2 3/4   ? */
184, 185, 186, 187, 188, 189, 190, 191, /*  A`  A'  A^  A~  A:  Ao  AE  C, */
 48,  48,  48,  48,  48,  48,  48,  48, /*  E`  E'  E^  E:  I`  I'  I^  I: */
 48,  48,  48,  48,  48,  48,  48,  48, /*  D-  N~  O`  O'  O^  O~  O:   X */
 48,  48,  48,  48,  48,  48,  48, 215, /*  O/  U`  U'  U^  U:  Y'   P   B */
 48,  48,  48,  48,  48,  48,  48,  48, /*  a`  a'  a^  a~  a:  ao  ae  c, */
 48,  48,  48,  48,  48,  48,  48,  48, /*  e`  e'  e^  e:  i`  i'  i^  i: */
 48,  48,  48,  48,  48,  48,  48,  48, /*  d   n~  o`  o'  o^  o~  o:  -: */
 48,  48,  48,  48,  48,  48,  48, 247, /*  o/  u`  u'  u^  u:  y'   P  y: */
 48,  48,  48,  48,  48,  48,  48,  48};
```

For example, the string ‘‘33:48,37:48,45-47:48,38:48’’ 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 useful for cutting and pasting electronic mailing addresses and filenames.

ACTIONS

It is possible to rebind keys (or sequences of keys) to arbitrary strings for input, by changing the **translations** resources for the vt100 or tek4014 widgets. Changing the **translations** resource for events other than key and button events is not expected, and will cause unpredictable behavior. The following actions are provided for use within the *vt100* or *tek4014* **translations** resources:

allow-send-events(*on/off/toggle***)**

This action set or toggles the **allowSendEvents** resource and is also invoked by the **allowsends** entry in *mainMenu*.

alt-sends-escape()

This action toggles the state of the **eightBitInput** resource.

bell([*percent***])**

This action rings the keyboard bell at the specified percentage above or below the base volume.

clear-saved-lines()

This action does **hard-reset()** (see below) and also clears the history of lines saved off the top of the screen. It is also invoked from the **clearsavedlines** entry in *vtMenu*. The effect is identical to a hardware reset (RIS) control sequence.

create-menu(*m/v/f/t***)**

This action creates one of the menus used by *xterm*, if it has not been previously created. The parameter values are the menu names: *mainMenu*, *vtMenu*, *fontMenu*, *tekMenu*, respectively.

dabbrev-expand()
Expands the word before cursor by searching in the preceding text on the screen and in the scroll-back buffer for words starting with that abbreviation. Repeating **dabbrev-expand()** several times in sequence searches for an alternative expansion by looking farther back. Lack of more matches is signaled by a **beep()**. Attempts to expand an empty word (i.e., when cursor is preceded by a space) yield successively all previous words. Consecutive identical expansions are ignored. The word here is defined as a sequence of non-whitespace characters. This feature partially emulates the behavior of 'dynamic abbreviation' expansion in Emacs (bound there to M-/). Here is a resource setting for *xterm* which will do the same thing: *VT100*translations: #override \n\
Meta <KeyPress> /:dabbrev-expand()

deiconify()
Changes the window state back to normal, if it was iconified.

delete-is-del()
This action toggles the state of the **deleteIsDEL** resource.

dired-button()
Handles a button event (other than press and release) by echoing the event's position (i.e., character line and column) in the following format:

ˆX ESC G <line+' '> <col+' '>

iconify()
Iconifies the window.

hard-reset()
This action resets the scrolling region, tabs, window size, and cursor keys and clears the screen. It is also invoked from the **hardreset** entry in *vtMenu*.

ignore() This action ignores the event but checks for special pointer position escape sequences.

insert() This action inserts the character or string associated with the key that was pressed.

insert-eight-bit()
This action inserts an eight-bit (Meta) version of the character or string associated with the key that was pressed. This only applies to single-byte values. The exact action depends on the value of the **metaSendsEscape** and the **eightBitInput** resources. The **metaSendsEscape** resource is tested first.

The term "eight-bit" is misleading: *xterm* checks if the key's value is less than 128. If so, *xterm* adds 128 to the value, setting its eighth bit. Otherwise *xterm* sends an ESC byte before the key. In other applications' documentation, that is referred to as a "meta key".

insert-selection(*sourcename* **[, ...])**
This action inserts the string found in the selection or cutbuffer 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*.

insert-seven-bit()
This action is a synonym for **insert()** The term "seven-bit" is misleading: it only implies that *xterm* does not try to add 128 to the key's value as in **insert-eight-bit()**.

interpret(*control-sequence***)**
Interpret the given control sequence locally, i.e., without passing it to the host. This works by inserting the control sequence at the front of the input buffer. Use "\" to escape octal digits in the string. Xt does not allow you to put a null character (i.e., "\000") in the string.

keymap(*name***)**
This action dynamically defines a new translation table whose resource name is *name* with the suffix *Keymap* (case is significant). The name *None* restores the original translation table.

larger-vt-font()
Set the font to the next larger one, based on the font dimensions. See also **set-vt-font()**.

load-vt-fonts(*name***[,***class***])**
Load fontnames from the given subresource name and class. That is, load the "*VT100.*name*.font", resource as "*VT100.font" etc. If no name is given, the original set of fontnames is restored.

Unlike **set-vt-font()**, this does not affect the escape- and select-fonts, since those are not based on resource values. It does affect the fonts loosely organized under the ‘‘Default’’ menu entry: **font**, **boldFont**, **wideFont** and **wideBoldFont**.

maximize()
Resizes the window to fill the screen.

meta-sends-escape()
This action toggles the state of the **metaSendsEscape** resource.

popup-menu(*menuname***)**
This action displays the specified popup menu. Valid names (case is significant) include: *mainMenu*, *vtMenu*, *fontMenu*, and *tekMenu*.

print() This action prints the window and is also invoked by the *print* entry in *mainMenu*.

print-redir()
This action toggles the **printerControlMode** between 0 and 2. The corresponding popup menu entry is useful for switching the printer off if you happen to change your mind after deciding to print random binary files on the terminal.

quit() This action sends a SIGHUP to the subprogram and exits. It is also invoked by the **quit** entry in *mainMenu*.

redraw()
This action redraws the window and is also invoked by the *redraw* entry in *mainMenu*.

restore()
Restores the window to the size before it was last maximized.

scroll-back(*count* **[,***units* **[,***mouse***]])**
This action 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.

An adjustment can be specified for these values by appending a "+" or "-" sign followed by a number, e.g., *page-2* to specify 2 lines less than a page.

If the third parameter *mouse* is given, the action is ignored when mouse reporting is enabled.

scroll-forw(*count* **[,***units* **[,***mouse***]])**
This action is similar to **scroll-back** except that it scrolls in the other direction.

secure() This action toggles the *Secure Keyboard* mode described in the section named **SECURITY**, and is invoked from the **securekbd** entry in *mainMenu*.

select-cursor-end(*destname* **[, ...])**
This action is similar to **select-end** except that it should be used with **select-cursor-start**.

select-cursor-extend()
This action is similar to **select-extend** except that it should be used with **select-cursor-start**.

select-cursor-start()
This action is similar to **select-start** except that it begins the selection at the current text cursor position.

select-end(*destname* **[, ...])**
This action puts the currently selected text into all of the selections or cutbuffers specified by *destname*.

select-extend()
This action tracks the pointer and extends the selection. It should only be bound to Motion events.

select-set()
This action stores text that corresponds to the current selection, without affecting the selection mode.

select-start()
This action begins text selection at the current pointer location. See the section on **POINTER USAGE** for information on making selections.

send-signal(*signame***)**
This action sends the signal named by *signame* 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*, *alrm*, *alarm* (same as *alrm*) and *kill*.

set-allow132(*on/off/toggle***)**
This action toggles the **c132** resource and is also invoked from the **allow132** entry in *vtMenu*.

set-altscreen(*on/off/toggle***)**
This action toggles between the alternate and current screens.

set-appcursor(*on/off/toggle***)**
This action toggles the handling Application Cursor Key mode and is also invoked by the **appcursor** entry in *vtMenu*.

set-appkeypad(*on/off/toggle***)**
This action toggles the handling of Application Keypad mode and is also invoked by the **appkeypad** entry in *vtMenu*.

set-autolinefeed(*on/off/toggle***)**
This action toggles automatic insertion of linefeeds and is also invoked by the **autolinefeed** entry in *vtMenu*.

set-autowrap(*on/off/toggle***)**
This action toggles automatic wrapping of long lines and is also invoked by the **autowrap** entry in *vtMenu*.

set-backarrow(*on/off/toggle***)**
This action toggles the **backarrowKey** resource and is also invoked from the **backarrow key** entry in *vtMenu*.

set-bellIsUrgent(*on/off/toggle***)**
This action toggles the **bellIsUrgent** resource and is also invoked by the **bellIsUrgent** entry in *vtMenu*.

set-cursorblink(*on/off/toggle***)**
This action toggles the **cursorBlink** resource and is also invoked from the **cursorblink** entry in *vtMenu*.

set-cursesemul(*on/off/toggle***)**
This action toggles the **curses** resource and is also invoked from the **cursesemul** entry in *vtMenu*.

set-font-doublesize(*on/off/toggle***)**
This action toggles the **fontDoublesize** resource and is also invoked by the **font-doublesize** entry in *fontMenu*.

set-hp-function-keys(*on/off/toggle***)**
This action toggles the **hpFunctionKeys** resource and is also invoked by the **hpFunctionKeys** entry in *mainMenu*.

set-jumpscroll(*on/off/toggle***)**
This action toggles the **jumpscroll** resource and is also invoked by the **jumpscroll** entry in *vtMenu*.

set-font-linedrawing(*on/off/toggle***)**
This action toggles the *xterm*'s state regarding whether the current font has line-drawing characters and whether it should draw them directly. It is also invoked by the **font-linedrawing** entry in *fontMenu*.

set-keep-selection(*on/off/toggle***)**
This action toggles the **keepSelection** resource and is also invoked by the **keepSelection** entry in *vtMenu*.

set-logging()
This action toggles the state of the logging option.

set-old-function-keys(*on/off/toggle***)**
This action toggles the state of legacy function keys and is also invoked by the **oldFunctionKeys** entry in *mainMenu*.

set-marginbell(*on/off/toggle***)**
This action toggles the **marginBell** resource.

set-num-lock()
This action toggles the state of the **numLock** resource.

set-pop-on-bell(*on/off/toggle***)**
This action toggles the **popOnBell** resource and is also invoked by the **poponbell** entry in *vtMenu*.

set-render-font(*on/off/toggle***)**
This action toggles the **renderFont** resource and is also invoked by the **render-font** entry in *fontMenu*.

set-reverse-video(*on/off/toggle***)**
This action toggles the **reverseVideo** resource and is also invoked by the **reversevideo** entry in *vtMenu*.

set-reversewrap(*on/off/toggle***)**
This action toggles the **reverseWrap** resource and is also invoked by the **reversewrap** entry in *vtMenu*.

set-scroll-on-key(*on/off/toggle***)**
This action toggles the **scrollKey** resource and is also invoked from the **scrollkey** entry in *vtMenu*.

set-scroll-on-tty-output(*on/off/toggle***)**
This action toggles the **scrollTtyOutput** resource and is also invoked from the **scrollttyoutput** entry in *vtMenu*.

set-scrollbar(*on/off/toggle***)**
This action toggles the **scrollbar** resource and is also invoked by the **scrollbar** entry in *vtMenu*.

set-select(*on/off/toggle***)**
This action toggles the **selectToClipboard** resource and is also invoked by the **selectToClipboard** entry in *vtMenu*.

set-sco-function-keys(*on/off/toggle***)**
This action toggles the **scoFunctionKeys** resource and is also invoked by the **scoFunctionKeys** entry in *mainMenu*.

set-sun-function-keys(*on/off/toggle***)**
This action toggles the **sunFunctionKeys** resource and is also invoked by the **sunFunctionKeys** entry in *mainMenu*.

set-sun-keyboard(*on/off/toggle***)**
This action toggles the **sunKeyboard** resource and is also invoked by the **sunKeyboard** entry in *mainMenu*.

set-tek-text(*large/2/3/small***)**
This action 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*.

set-terminal-type(*type***)**
This action 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-titeInhibit(*on/off/toggle***)**
This action toggles the **titeInhibit** resource, which controls switching between the alternate and current screens.

set-toolbar(*on/off/toggle***)**
This action toggles the toolbar feature and is also invoked by the **toolbar** entry in *mainMenu*.

set-utf8-mode(*on/off/toggle***)**
This action toggles the **utf8** resource and is also invoked by the **utf8-mode** entry in *fontMenu*.

set-utf8-title(*on/off/toggle***)**
This action toggles the **utf8Title** resource and is also invoked by the **utf8-title** entry in *fontMenu*.

set-visibility(*vt/tek***,***on/off/toggle***)**
This action 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-visual-bell(*on/off/toggle***)**
This action toggles the **visualBell** resource and is also invoked by the **visualbell** entry in *vtMenu*.

set-vt-font(*d/1/2/3/4/5/6/e/s* **[,***normalfont* **[,** *boldfont***]])**
This action 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* indicate 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* indicate the normal and bold fonts that have been set through escape codes (or specified as the second and third action arguments, respectively), and

s or *S* indicate the font selection (as made by programs such as *xfontsel(1)*) indicated by the second action argument.

If *xterm* is configured to support wide characters, an additional two optional parameters are recognized for the *e* argument: wide font and wide bold font.

smaller-vt-font()
Set the font to the next smaller one, based on the font dimensions. See also **set-vt-font()**.

soft-reset()
This action resets the scrolling region and is also invoked from the **softreset** entry in *vtMenu*. The effect is identical to a soft reset (DECSTR) control sequence.

spawn-new-terminal(params)
Spawn a new *xterm* process. This is available on systems which have a modern version of the process filesystem, e.g., "/proc", which *xterm* can read.

Use the "cwd" process entry, e.g., /proc/12345/cwd to obtain the working directory of the process which is running in the current *xterm*.

On systems which have the "exe" process entry, e.g., /proc/12345/exe, use this to obtain the actual executable. Otherwise, use the $PATH variable to find *xterm*.

If parameters are given in the action, pass them to the new *xterm* process.

start-extend()
This action is similar to **select-start** except that the selection is extended to the current pointer location.

start-cursor-extend()
This action is similar to **select-extend** except that the selection is extended to the current text cursor position.

string(*string***)**
This action 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.

tek-copy()
This action 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*.

tek-page()
This action clears the Tektronix window and is also invoked by the **tekpage** entry in *tekMenu*.

tek-reset()
This action resets the Tektronix window and is also invoked by the *tekreset* entry in *tekMenu*.

vi-button()
Handles a button event (other than press and release) by echoing a control sequence computed from the event’s line number in the screen relative to the current line:

ESC ^P

or

ESC ^N

according to whether the event is before, or after the current line, respectively. The ^N (or ^P) is repeated once for each line that the event differs from the current line. The control sequence is omitted altogether if the button event is on the current line.

visual-bell()
This action flashes the window quickly.

The Tektronix window also has the following action:

gin-press(*l/L/m/M/r/R***)**
This action sends the indicated graphics input code.

The default bindings in the VT102 window use the SELECT token, which is set by the **selectToClipboard** resource:

```
        Shift <KeyPress> Prior:scroll-back(1,halfpage) \n\
         Shift <KeyPress> Next:scroll-forw(1,halfpage) \n\
       Shift <KeyPress> Select:select-cursor-start() \
                        select-cursor-end(SELECT, CUT_BUFFER0) \n\
      Shift <KeyPress> Insert:insert-selection(SELECT, CUT_BUFFER0) \n\
        <KeyPress> XF86Paste:insert-selection(SELECT, CUT_BUFFER0) \n\
         <KeyPress> SunPaste:insert-selection(SELECT, CUT_BUFFER0) \n\
   Shift~Ctrl <KeyPress> KP_Add:larger-vt-font() \n\
```

```
    Shift Ctrl <KeyPress> KP_Add:smaller-vt-font() \n\
    Shift <KeyPress> KP_Subtract:smaller-vt-font() \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\
  !Lock Ctrl @Num_Lock <Btn1Down>:popup-menu(mainMenu) \n\
    ! @Num_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\
  !Lock Ctrl @Num_Lock <Btn2Down>:popup-menu(vtMenu) \n\
    ! @Num_Lock Ctrl <Btn2Down>:popup-menu(vtMenu) \n\
        ~Ctrl ~Meta <Btn2Down>:ignore() \n\
              Meta <Btn2Down>:clear-saved-lines() \n\
         ~Ctrl ~Meta <Btn2Up>:insert-selection(SELECT, CUT_BUFFER0) \n\
             !Ctrl <Btn3Down>:popup-menu(fontMenu) \n\
         !Lock Ctrl <Btn3Down>:popup-menu(fontMenu) \n\
  !Lock Ctrl @Num_Lock <Btn3Down>:popup-menu(fontMenu) \n\
    ! @Num_Lock Ctrl <Btn3Down>:popup-menu(fontMenu) \n\
        ~Ctrl ~Meta <Btn3Down>:start-extend() \n\
          ~Meta <Btn3Motion>:select-extend() \n\
            Ctrl <Btn4Down>:scroll-back(1,halfpage,m) \n\
        Lock Ctrl <Btn4Down>:scroll-back(1,halfpage,m) \n\
 Lock @Num_Lock Ctrl <Btn4Down>:scroll-back(1,halfpage,m) \n\
     @Num_Lock Ctrl <Btn4Down>:scroll-back(1,halfpage,m) \n\
               <Btn4Down>:scroll-back(5,line,m)     \n\
           Ctrl <Btn5Down>:scroll-forw(1,halfpage,m) \n\
        Lock Ctrl <Btn5Down>:scroll-forw(1,halfpage,m) \n\
 Lock @Num_Lock Ctrl <Btn5Down>:scroll-forw(1,halfpage,m) \n\
     @Num_Lock Ctrl <Btn5Down>:scroll-forw(1,halfpage,m) \n\
               <Btn5Down>:scroll-forw(5,line,m)     \n\
                 <BtnUp>:select-end(SELECT, CUT_BUFFER0) \n\
                <BtnDown>:ignore()
```

The default bindings for the scrollbar widget are separate from the VT100 widget:

```
               <Btn5Down>: StartScroll(Forward) \n\
               <Btn1Down>: StartScroll(Forward) \n\
               <Btn2Down>: StartScroll(Continuous) MoveThumb() NotifyThumb() \n\
               <Btn3Down>: StartScroll(Backward) \n\
               <Btn4Down>: StartScroll(Backward) \n\
               <Btn2Motion>: MoveThumb() NotifyThumb() \n\
               <BtnUp>:    NotifyScroll(Proportional) EndScroll()
```

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\
 !Lock Ctrl @Num_Lock <Btn1Down>: popup-menu(mainMenu) \n\
    !Ctrl @Num_Lock <Btn1Down>: popup-menu(mainMenu) \n\
             !Ctrl <Btn2Down>: popup-menu(tekMenu) \n\
         !Lock Ctrl <Btn2Down>: popup-menu(tekMenu) \n\
 !Lock Ctrl @Num_Lock <Btn2Down>: popup-menu(tekMenu) \n\
    !Ctrl @Num_Lock <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)

Here is an example which uses shifted select/paste to copy to the clipboard, and unshifted select/paste for the primary selection. In each case, a (different) cut buffer is also a target or source of the select/paste operation. It is important to remember however, that cut buffers store data in ISO-8859-1 encoding, while selections can store data in a variety of formats and encodings. While *xterm* owns the selection, it highlights it. When it loses the selection, it removes the corresponding highlight. But you can still paste from the corresponding cut buffer. *VT100*translations: #override \n\
~Shift~Ctrl<Btn2Up>: insert-selection(PRIMARY, CUT_BUFFER0) \n\
Shift~Ctrl<Btn2Up>: insert-selection(CLIPBOARD, CUT_BUFFER1) \n\
~Shift<BtnUp>: select-end(PRIMARY, CUT_BUFFER0) \n\
Shift<BtnUp>: select-end(CLIPBOARD, CUT_BUFFER1)

Below is a sample how of the **keymap()** action is used to add special keys for entering commonly-typed works: *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)

Some people prefer using the left pointer button for dragging the scrollbar thumb. That can be setup by altering the translations resource, e.g., *VT100.scrollbar.translations:#override \n\ <Btn5Down>: StartScroll(Forward) \n\ <Btn1Down>: StartScroll(Continuous) MoveThumb() NotifyThumb() \n\ <Btn4Down>: StartScroll(Backward) \n\ <Btn1Motion>: MoveThumb() NotifyThumb() \n\ <BtnUp>: NotifyScroll(Proportional) EndScroll()

CONTROL SEQUENCES AND KEYBOARD

The *Xterm Control Sequences* document lists the control sequences which an application can send *xterm* to make it perform various operations. Most of these operations are standardized, from either the DEC or Tektronix terminals, or from more widely used standards such as ISO-6429.

ENVIRONMENT

Xterm sets several environment variables:

DISPLAY
is the display name, pointing to the X server (see **DISPLAY NAMES** in X()).

TERM
is set according to the termcap (or terminfo) entry which it is using as a reference.

WINDOWID
is set to the X window id number of the *xterm* window.

XTERM_LOCALE
shows the locale which was used by *xterm* on startup. Some shell initialization scripts may set a different locale.

XTERM_SHELL
is set to the pathname of the program which is invoked. Usually that is a shell program, e.g., **/bin/sh**. Since it is not necessarily a shell program however, it is distinct from ‘‘SHELL’’.

XTERM_VERSION
is set to the string displayed by the **-version** option. That is normally an identifier for the X Window libraries used to build *xterm*, followed by *xterm*’s patch number in parenthesis. The patch number is also part of the response to a Secondary Device Attributes (DA) control sequence (see *Xterm Control Sequences*).

Depending on your system configuration, *xterm* may also set the following:

COLUMNS
the width of the *xterm* in characters (cf: "stty columns").

HOME
when *xterm* is configured to update utmp.

LINES
the height of the *xterm* in characters (cf: "stty rows").

LOGNAME
when *xterm* is configured to update utmp.

SHELL
when *xterm* is configured to update utmp. It is also set if you provide the shell name as the optional parameter.

TERMCAP
the contents of the termcap entry corresponding to $TERM, with lines and columns values substituted for the actual size window you have created.

TERMINFO
may be defined to a nonstandard location in the configure script.

FILES

The actual pathnames given may differ on your system.

/var/run/utmp
the system logfile, which records user logins.

/var/log/wtmp
the system logfile, which records user logins and logouts.

/etc/X11/app-defaults/XTerm
the *xterm* default application resources.

/etc/X11/app-defaults/XTerm-color
the *xterm* color application resources. If your display supports color, use this

```
*customization: -color
```

in your .Xdefaults file to automatically use this resource file rather than */etc/X11/app-defaults/XTerm*. If you do not do this, *xterm* uses its compiled-in default resource settings for colors.

ERROR MESSAGES

Most of the fatal error messages from *xterm* use the following format:

xterm: Error *XXX*, errno *YYY*: *ZZZ*

The *XXX* codes (which are used by *xterm* as its exit-code) are listed below, with a brief explanation.

1 is used for miscellaneous errors, usually accompanied by a specific message,

11 ERROR_FIONBIO
main: ioctl() failed on FIONBIO

12 ERROR_F_GETFL
main: ioctl() failed on F_GETFL

13 ERROR_F_SETFL
main: ioctl() failed on F_SETFL

14 ERROR_OPDEVTTY
spawn: open() failed on /dev/tty

15 ERROR_TIOCGETP
spawn: ioctl() failed on TIOCGETP

17 ERROR_PTSNAME
spawn: ptsname() failed

18 ERROR_OPPTSNAME
spawn: open() failed on ptsname

19 ERROR_PTEM
spawn: ioctl() failed on I_PUSH/"ptem"

20 ERROR_CONSEM
spawn: ioctl() failed on I_PUSH/"consem"

21 ERROR_LDTERM
spawn: ioctl() failed on I_PUSH/"ldterm"

22 ERROR_TTCOMPAT
spawn: ioctl() failed on I_PUSH/"ttcompat"

23 ERROR_TIOCSETP
spawn: ioctl() failed on TIOCSETP

24 ERROR_TIOCSETC
spawn: ioctl() failed on TIOCSETC

25 ERROR_TIOCSETD
spawn: ioctl() failed on TIOCSETD

26 ERROR_TIOCSLTC
spawn: ioctl() failed on TIOCSLTC

27 ERROR_TIOCLSET
spawn: ioctl() failed on TIOCLSET

28 ERROR_INIGROUPS
spawn: initgroups() failed

29 ERROR_FORK
spawn: fork() failed

30 ERROR_EXEC
spawn: exec() failed

32 ERROR_PTYS
get_pty: not enough ptys

34 ERROR_PTY_EXEC
waiting for initial map

35 ERROR_SETUID
spawn: setuid() failed

36 ERROR_INIT
spawn: can’t initialize window

46 ERROR_TIOCKSET
spawn: ioctl() failed on TIOCKSET

47 ERROR_TIOCKSETC
spawn: ioctl() failed on TIOCKSETC

48 ERROR_SPREALLOC
spawn: realloc of ttydev failed

49 ERROR_LUMALLOC
luit: command-line malloc failed

50 ERROR_SELECT
in_put: select() failed

54 ERROR_VINIT
VTInit: can’t initialize window

57 ERROR_KMMALLOC1
HandleKeymapChange: malloc failed

60 ERROR_TSELECT
Tinput: select() failed

64 ERROR_TINIT
TekInit: can’t initialize window

71 ERROR_BMALLOC2
SaltTextAway: malloc() failed

80 ERROR_LOGEXEC
StartLog: exec() failed

83 ERROR_XERROR
xerror: XError event

84 ERROR_XIOERROR
xioerror: X I/O error

90 ERROR_SCALLOC
Alloc: calloc() failed on base

91 ERROR_SCALLOC2
Alloc: calloc() failed on rows

92 ERROR_SREALLOC
ScreenResize: realloc() failed on alt base

96 ERROR_RESIZE
ScreenResize: malloc() or realloc() failed

102 ERROR_SAVE_PTR
ScrnPointers: malloc/realloc() failed

110 ERROR_SBRALLOC
ScrollBarOn: realloc() failed on base

111 ERROR_SBRALLOC2
ScrollBarOn: realloc() failed on rows

121 ERROR_MMALLOC
my_memmove: malloc/realloc failed

BUGS

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 pty 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.

This program still needs to be rewritten. It should be split into very modular sections, with the various emulators being completely separate widgets that do not 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 Tek COPY file name.

SEE ALSO

resize(1), luit(1), X(), pty(4), tty(4)
Xterm Control Sequences (this is the file ctlseqs.ms).

http://invisible-island.net/xterm/xterm.html

http://invisible-island.net/xterm/ctlseqs/ctlseqs.html

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), Jonathan Kamens (MIT-Athena), Jason Bacon, Stephen P. Wall, David Wexelblat, and Thomas Dickey (invisible-island.net).