



Full credit is given to the above companies including the OS that this PDF file was generated!

Rocky Enterprise Linux 9.2 Manual Pages on command 'xterm.1'

\$ man xterm.1

XTERM(1) X Window System XTERM(1)

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 and selected features from higher-level terminals such as VT320/VT420/VT520 (VTxxx). It also provides Tektronix 4014 emulation 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

Xterm provides usable emulations of related DEC terminals:

? VT52 emulation is complete.

? VT102 emulation is fairly complete, but does not support autorepeat (because that would affect the keyboard used by other X clients).

Double-size characters are displayed properly if your font server supports scalable bitmap fonts.

? VT220 emulation does not support soft fonts, it is otherwise complete.

? VT420 emulation (the default) supports controls for manipulating rectangles of characters as well as left/right margins.

Xterm does not support some other features which are not suitable for emulation, e.g., two-sessions.

Terminal database (terminfo (5) or termcap (5)) entries that work with xterm include

an optional platform-specific entry (?xterm?),

?xterm?,

?vt102?,

?vt100?,

?ansi? and

?dumb?

Xterm automatically searches the terminal database in this order for these entries and then sets the ?TERM? variable (and the ?TERMCAP? environment variable on a few older systems). The alternatives after ?xterm? are very old, from the late 1980s.

VT100 and VT102 emulations are commonly equated, though they actually differ. The VT102 provided controls for inserting and deleting lines.

Similarly, `?ansi?` and `?vt100?` are often equated. These are not really the same. For instance, they use different controls for scrolling (but `xterm` supports both). These features differ in an `?ansi?` terminal description from `xterm`:

`acsc`

Pseudo-graphics (line-drawing) uses a different mapping.

`xenl`

Xterm wraps text at the right margin using the VT100 `?newline glitch?` behavior.

Because of the wrapping behavior, you would occasionally have to repaint the screen when using a text editor with the `?ansi?` description.

You may also use descriptions corresponding to the various supported emulations such as `?vt220?` or `?vt420?`, but should set the terminal emulation level with the `decTerminalID` resource.

On most systems, `xterm` will use the terminfo database. Some older systems use `termcap`. (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 VTxxx escape sequences (see Xterm Control Sequences).

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

`?COPYyyyy-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 VTxxx 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 usual terminal description 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 VTxxx or Tektronix mode, there are escape sequences to change the name of the windows. Additionally, in VTxxx 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

Because xterm uses the X Toolkit library, it accepts the standard X Toolkit command line options. Xterm also accepts many application-

specific options.

By convention, if an option begins with a `?+?` instead of a `?-?`, the option is restored to its default value.

Most of the xterm options are actually parsed by the X Toolkit, which sets resource values, and overrides corresponding resource-settings in your X resource files. Xterm provides the X Toolkit with a table of options. A few of these are marked, telling the X Toolkit to ignore them (`-help`, `-version`, `-class`, `-e`, and `-into`). After the X Toolkit has parsed the command-line parameters, it removes those which it handles, leaving the specially-marked parameters for xterm to handle.

These options do not set a resource value, and are handled specially:

`-version`

This causes xterm to print a version number to the standard output, and then exit.

`-help` This causes xterm to print out a verbose message describing its

options, one per line. The message is written to the standard output. After printing the message, xterm exits. 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.

The `-version` and `-help` options are interpreted even if xterm cannot open the display, and are useful for testing and configuration scripts.

Along with `-class`, they are checked before other options. To do this, xterm has its own (much simpler) argument parser, along with a table of the X Toolkit's built-in list of options.

Relying upon the X Toolkit to parse the options and associated values has the advantages of simplicity and good integration with the X

resource mechanism. There are a few drawbacks

? Xterm cannot tell easily whether a resource value was set by one of the external resource- or application-defaults files, whether it was set using `xrdb(1)`, or if it was set through the `-xrm` option or via some directly relevant command-line option. Xterm sees only the end-result: a value supplied when creating its widgets.

? Xterm does not know the order in which particular options and items in resource files are evaluated. Rather, it sees all of the values for a given widget at the same time. In the design of these options, some are deemed more important, and can override other options.

The X Toolkit uses patterns (constants and wildcards) to match resources. Once a particular pattern has been used, it will not modify it. To override a given setting, a more-specific pattern must be used, e.g., replacing `?*` with `?.*`. Some poorly-designed resource files are too specific to allow the command-line options to affect the relevant widget values.

? In a few cases, the X Toolkit combines its standard options in ways which do not work well with xterm. This happens with the color (`-fg`, `-bg`) and reverse (`-rv`) options. Xterm makes a special case of these and adjusts its sense of `?reverse?` to lessen user surprise.

One parameter (after all options) may be given. That overrides xterm's built-in choice of shell program:

? If the parameter is not a relative path, i.e., beginning with `?./?` or `?../?`, xterm looks for the file in the user's `PATH`. In either case, this check fails if xterm cannot construct an absolute path.

? If that check fails (or if no such parameter is given), xterm next checks the `?SHELL?` variable. If that specifies an executable file, xterm will attempt to start that. However, xterm additionally checks if it is a valid shell, and will unset `?SHELL?` if it is not.

? If `?SHELL?` is not set to an executable file, xterm tries to use the shell program specified in the user's password file entry. As

before, xterm verifies if this is a valid shell.

? Finally, if the password file entry does not specify a valid shell, xterm uses /bin/sh.

The -e option cannot be used with this parameter since it uses all parameters following the option.

Xterm validates shell programs by finding their pathname in the text file /etc/shells. It treats the environment variable ?SHELL? specially because (like ?TERM?), xterm both reads and updates the variable, and because the program started by xterm is not necessarily a shell.

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, and is equivalent to setting the vt100 resource autoWrap to ?true?.

Auto-wraparound 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, and is equivalent to setting the vt100 resource autoWrap to ?false?.

-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?.

-baudrate number

Set the line-speed, used to test the behavior of applications that use the line-speed when optimizing their output to the screen. The default is ?38400?.

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

+bc turn off 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 cursorOnTime 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 CHARACTER CLASSES and 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.

X Toolkit sets the `WM_CLASS` property using the instance name and this class value.

`-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.

NOTE: 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.

-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.

If you specify both -fa and the X Toolkit option -fn, the -fa

setting overrides the latter.

See also the `renderFont` resource, which combines with this to determine whether FreeType fonts are initially active.

`-fb font`

This option specifies a font to be used when displaying bold text. It sets the `boldFont` resource.

This font must be the same height and width as the normal font, otherwise it is ignored. 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.

See also the discussion of `boldMode` and `alwaysBoldMode` resources.

`-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?`.

`-fc fontchoice`

Specify the initial font chosen from the font menu. The option value corresponds to the `initialFont` resource.

`-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.

-fullscreen

This option indicates that xterm should ask the window manager to let it use the full-screen for display, e.g., without window decorations. It sets the fullscreen resource to ?true?.

+fullscreen

This option indicates that xterm should not ask the window manager to let it use the full-screen for display. It sets the fullscreen resource to ?false?.

-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 double-width 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 double-width 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. (This option is ignored on most systems, because TERMCAP is not used).
- +im Turn off the useInsertMode resource.
- into windowId

Given an X window identifier (an integer, which can be hexadecimal, octal or decimal according to whether it begins

with "0x", "0" or neither), xterm will reparent its top-level shell widget to that window. This is used to embed xterm within other applications.

For instance, there are scripts for Tcl/Tk and Gtk which can be used to demonstrate the feature. When using Gtk, there is a limitation of that toolkit which requires that xterm's allowSendEvents resource is enabled.

- itc Set the vt100 resource colorITMode to ?false?, disabling the display of characters with italic attribute as color.
- +itc Set the vt100 resource colorITMode to ?true?, enabling the display of characters with italic attribute as color rather than italic.
- 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?, ?legacy?, ?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,
oldXtermFKeys and
sunKeyboard,

using the Sun/PC keyboard layout.

- l Turn logging on, unless disabled by the logInhibit resource.
Some versions of xterm may have logging enabled. However, normally logging is not supported, due to security concerns in the early 1990s. That was a problem in X11R4 xterm (1989) which was addressed by a patch to X11R5 late in 1993. X11R6 included these fixes. The older version (when running with root privilege) would create the log file using root privilege. The reason why xterm ran with root privileges was to open pseudo-terminals. Those privileges are now needed only on very old systems: Unix98 pseudo-terminals made the BSD scheme unnecessary.

Unless overridden by the -lf option or the logFile resource:

? If the filename is ?-?, then logging is sent to the standard output.

? Otherwise a filename is generated, and the log file is written to the directory from which xterm is invoked.

? The generated filename is 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. This sets the logFile resource. If set to ?-?, xterm writes its log to the standard output. 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 `/var/log/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?`).

`-maximized`

This option indicates that `xterm` should ask the window manager to maximize its layout on startup. This corresponds to the maximized resource.

Maximizing is not the reverse of iconifying; it is possible to do both with certain window managers.

`+maximized`

This option indicates that `xterm` should ask the window manager to not maximize its layout on startup.

`-mb` This option indicates that `xterm` should ring a margin bell when the user types near the right end of a line.

`+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.

-pf font

This option specifies the font to be used for the pointer. The corresponding resource name is pointerFont. The resource value default is cursor.

-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.

-report-charclass

Print a report to the standard output showing information about the character-classes which can be altered using the charClass resource.

-report-colors

Print a report to the standard output showing information about colors as xterm allocates them. This corresponds to the reportColors resource.

-report-fonts

Print a report to the standard output showing information about fonts which are loaded. This corresponds to the reportFonts resource.

-report-icons

Print a report to the standard output showing information about pixmap-icons which are loaded. This corresponds to the reportIcons resource.

-report-xres

Print a report to the standard output showing the values of boolean, numeric or string X resources for the VT100 widget when initialization is complete. This corresponds to the reportXRes resource.

-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.

-sh number

scale line-height values by the given number. See the discussion of the scaleHeight resource.

-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 ?1024?.

-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 VTxxx mode. Switching between the two windows is done using the ?Options? menus.

Terminal database (terminfo (5) or termcap (5)) entries that work with xterm are:

?tek4014?,

?tek4015?,

?tek4012?,

?tek4013?,

?tek4010?, and

?dumb?.

Xterm automatically searches the terminal database in this order for these entries and then sets the ?TERM? variable (and the ?TERMCAP? environment variable, if relevant).

+t This option indicates that xterm should start in VTxxx 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, vt220, and vt240 (the ?vt? is optional). The default is ?vt420?. 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 UTF-8 encoding on and off, use the -wc option or the corresponding wideChars resource, rather than the -u8 option. 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.
- uc This option makes the cursor underlined instead of a box.
- +uc This option makes the cursor a box instead of underlined.
- 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 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 xterm is not started in UTF-8 mode (or if this resource is not set), initially it maintains those structures to support 8-bit characters. Xterm can later be switched, using a menu entry or control sequence, causing it to

reallocate those structures to support 16-bit characters.

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.

`-Scn` 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 (the first two are equivalent since the descriptor follows the last ?/?):

-S/dev/pts/123/45

-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.

Old Options

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`. The default icon name is the application name.

If no suitable icon is found, xterm provides a compiled-in pixmap.

X Toolkit sets the `WM_ICON_NAME` property using this value.

-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`.

X Toolkit Options

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. The corresponding resource name is `borderColor`. `Xterm` uses the X Toolkit default, which is `?XtDefaultForeground?`.

`Xterm`'s VT100 window has two borders: the inner border `internalBorder` and the outer border `borderWidth`, managed by the X Toolkit.

Normally `xterm` fills the inner border using the VT100 window's background color. If the `colorInnerBorder` resource is enabled, then `xterm` may fill the inner border using the `borderColor` resource.

`-bg color`

This option specifies the color to use for the background of the window. The corresponding resource name is `background`. 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(7).

-fg color

This option specifies the color to use for displaying text.

The corresponding resource name is foreground. The default is

?XtDefaultForeground?.

-fn font

This option specifies the font to be used for displaying normal text. The corresponding resource name is font. The resource value default is fixed.

-font font

This is the same as -fn.

-geometry geometry

This option specifies the preferred size and position of the VTxxx window; see X(7).

The normal geometry specification can be suffixed with @ followed by a Xinerama screen specification; it can be either g for the global screen (default), c for the current screen or a screen number.

-iconic

This option indicates that xterm should ask the window manager to start it as an icon rather than as the normal window. The corresponding resource name is iconic.

-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. The corresponding resource name is reverseVideo.

+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. It is shorthand for specifying the ?title? resource. The default title is the command line specified after the -e option, if any, otherwise the application name.

X Toolkit sets the WM_NAME property using this value.

-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.

X Toolkit accepts alternate names for a few of these options, e.g.,

? ?-background? for ?-bg?

? ?-font? for ?-fn?

? ?-foreground? for ?-fg?

Abbreviated options also are supported, e.g., ?-v? for ?-version.?

RESOURCES

Xterm understands all of the core X Toolkit resource names and classes.

It also uses the X Toolkit resource types (such as booleans, colors, fonts, integers, and strings) along with their respective converters.

Those resource types are not always sufficient:

? Xterm's resource values may be lists of names. X Toolkit resource types do not include lists. Xterm uses a string for the resource, and parses it.

Comma-separated lists of names ignore case.

? Xterm may defer processing a resource until it is needed. For example, font2 through font7 are loaded as needed, to start faster. Again, the actual resource type is a string, parsed and used when needed.

Application specific resources (e.g., ?XTerm.NAME?) follow:

Application Resources

backarrowKeyIsErase (class BackarrowKeyIsErase)

Tie the VTxxx backarrowKey and ptyInitialErase resources

together by setting the DECBKM state according to whether the initial erase character is a backspace (8) or delete (127) character. A `?false?` value disables this feature. The default is `?False?`.

Here are tables showing how the initial settings for

- ? `backarrowKeyIsErase` (BKIE),
- ? `backarrowKey` (BK), and
- ? `ptyInitialErase` (PIE), along with the
- ? stty erase character (^H for backspace, ^? for delete)

will affect DECBKM. First, xterm obtains the initial erase character:

- ? xterm's internal value is ^H
- ? xterm asks the operating system for the value which stty shows
- ? the `ttyModes` resource may override erase
- ? if `ptyInitialErase` is false, xterm will look in the terminal database

Summarizing that as a table:

PIE	stty	termcap	erase
????????????????????????????????????			
false	^H	^H	^H
false	^H	^?	^?
false	^?	^H	^H
false	^?	^?	^?
true	^H	^H	^H
true	^H	^?	^H
true	^?	^H	^?
true	^?	^?	^?

Using that erase character, xterm allows further choices:

- ? if `backarrowKeyIsErase` is true, xterm uses the erase character for the initial state of DECBKM
- ? if `backarrowKeyIsErase` is false, xterm sets DECBKM to 2 (internal). This ties together `backarrowKey` and the

control sequence for DECBKM.

? applications can send a control sequence to set/reset

DECBKM control set

? the ?Backarrow Key (BS/DEL)? menu entry toggles DECBKM

Summarizing the initialization details:

erase BKIE BK DECBKM result

??

^?	false	false	2	^H
^?	false	true	2	^?
^?	true	false	0	^?
^?	true	true	1	^?
^H	false	false	2	^H
^H	false	true	2	^?
^H	true	false	0	^H
^H	true	true	1	^H

buffered (class Buffered)

Normally xterm is built with double-buffer support. This resource can be used to turn it on or off. Setting the resource to ?true? turns double-buffering on. The default value is ?False?.

bufferedFPS (class BufferedFPS)

When xterm is built with double-buffer support, this gives the maximum number of frames/second. The default is ?40? and is limited to the range 1 through 100.

fullscreen (class Fullscreen)

Specifies whether or not xterm should ask the window manager to use a fullscreen layout on startup. Xterm accepts either a keyword (ignoring case) or the number shown in parentheses:

false (0)

Fullscreen layout is not used initially, but may be later via menu-selection or control sequence.

true (1)

Fullscreen layout is used initially, but may be disabled

later via menu-selection or control sequence.

always (2)

Fullscreen layout is used initially, and cannot be disabled

later via menu-selection or control sequence.

never (3)

Fullscreen layout is not used, and cannot be enabled later

via menu-selection or control sequence.

The default is ?false?.

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. The default is ?false?, i.e.,

this feature is disabled.

The keyboardType resource is the preferred mechanism for selecting this mode.

iconGeometry (class IconGeometry)

Specifies the preferred size and position of the application when iconified. It is not necessarily obeyed by all window managers.

iconHint (class IconHint)

Specifies an icon which will be added to the window manager hints. Xterm provides no default value.

Set this resource to ?none? to omit the hint entirely, using whatever the window manager may decide.

If the iconHint resource is given (or is set via the -n option) xterm searches for a pixmap file with that name, in the current

directory as well as in /usr/share/pixmaps. If the resource does not specify an absolute pathname. In each case, xterm adds ?_48x48? and/or ?.xpm? to the filename after trying without those suffixes. If it is able to load the file, xterm sets the window manager hint for the icon-pixmap. These pixmaps are distributed with xterm, and can optionally be compiled-in:

- ? mini.xterm_16x16, mini.xterm_32x32, mini.xterm_48x48
- ? filled-xterm_16x16, filled-xterm_32x32, filled-xterm_48x48
- ? xterm_16x16, xterm_32x32, xterm_48x48
- ? xterm-color_16x16, xterm-color_32x32, xterm-color_48x48

In either case, xterm allows for adding a ?_48x48? to specify the largest of the pixmaps as a default. That is, ?mini.xterm? is the same as ?mini.xterm_48x48?.

If no explicit iconHint resource is given (or if none of the compiled-in names matches), xterm uses ?mini.xterm? (which is always compiled-in).

The iconHint resource has no effect on ?desktop? files, including ?panel? and ?menu?. Those are typically set via a ?.desktop? file; xterm provides samples for itself (and the uxterm script). The more capable desktop systems allow changing the icon on a per-user basis.

iconName (class IconName)

Specifies a label for xterm when iconified. Xterm provides no default value; some window managers may assume the application name, e.g., ?xterm?.

Setting the iconName resource sets the icon label unless overridden by zIconBeep or the control sequences which change the window and icon labels.

keyboardType (class KeyboardType)

Enables one (or none) of the various keyboard-type resources:

hpFunctionKeys, scoFunctionKeys, sunFunctionKeys,
tcapFunctionKeys, oldXtermFKeys and sunKeyboard.

The resource's value should be one of the corresponding strings ?hp?, ?sco?, ?sun?, ?tcap?, ?legacy? or ?vt220?, respectively.

The individual resources are provided for legacy support; this resource is simpler to use. Xterm will use only one keyboard-type, but if multiple resources are set, it warns and uses the last one it checks.

The default is ?unknown?, i.e., none of the associated resources are set via this resource.

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.

maximized (class Maximized)

Specifies whether or not xterm should ask the window manager to maximize its layout on startup. The default is ?false?.

menuHeight (class MenuHeight)

Specifies the height of the toolbar, which may be increased by the X toolkit layout widget depending upon the fontsize used.

The default is ?25?.

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 ?C? (POSIX).

To use the current locale (only useful if you have localized the resource settings for the menu entries), set the resource to an empty string.

messages (class Messages)

Specifies whether write access to the terminal is allowed initially. See `mesg(1)`. The default is `?true?`.

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.

omitTranslation (class OmitTranslation)

Selectively omit one or more parts of `xterm`'s default translations at startup. The resource value is a comma-separated list of keywords, which may be abbreviated:

default

ignore (mouse) button-down events which were not handled by other translations

fullscreen

assigns a key-binding to the `fullscreen()` action.

keypress

assigns keypresses by default to the `insert-seven-bit()` and `insert-eight-bit()` actions.

scroll-back() and

`scroll-forw()` actions.

pointer

assigns pointer motion and button events to the `pointer-motion()` and `pointer-button()` actions respectively.

popup-menu

assigns mouse-buttons with the control modifier to the `popup-menus`.

reset assigns mouse-button 2 with the meta modifier to the

`clear-saved-lines` action.

scroll-lock

assigns a key-binding to the `scroll-lock()` action.

select assigns mouse- and keypress-combinations to actions

which manipulate the selection.

Xterm also uses these actions to capture mouse button and motion events which can be manipulated with the mouse protocol control sequences. If the select translations are omitted, then the pointer-motion and pointer-button handle these mouse protocol control sequences instead.

shift-fonts

assigns key-bindings to larger-vt-font() and smaller-vt-font() actions.

wheel-mouse

assigns buttons 4 and 5 with different modifiers to the scroll-back() and scroll-forw() actions.

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, if the system uses TERMCAP.

See also the ttyModes resource, which may override 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 initial 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 macOS systems, `?true?` otherwise.

`reportColors` (class `ReportColors`)

If true, xterm will print to the standard output a summary of colors as it allocates them. The default is `?false?`.

`reportFonts` (class `ReportFonts`)

If true, xterm will print to the standard output a summary of each font's metrics (size, number of glyphs, etc.), as it loads them. The default is `?false?`.

`reportIcons` (class `ReportIcons`)

If true, xterm will print to the standard output a summary of each pixmap icon as it loads them. The default is `?false?`.

`reportXRes` (class `ReportXRes`)

If true, xterm will print to the standard output a list of the boolean, numeric and string X resources for the VT100 widget after initialization. The default is `?false?`.

`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?`.

`scaleHeight` (class `ScaleHeight`)

Scale line-height values by the resource value, which is limited to `?0.9?` to `?1.5?`. The default value is `?1.0?`,

While this resource applies to either bitmap or TrueType fonts,

its main purpose is to help work around incompatible changes in the Xft library's font metrics. Xterm checks the font metrics to find what the library claims are the bounding boxes for each glyph (character). However, some of Xft's features (such as the autohinter) can cause the glyphs to be scaled larger than the bounding boxes, and be partly overwritten by the next row. See [useClipping](#) for a related resource.

scoFunctionKeys (class ScoFunctionKeys)

Specifies whether or not SCO function key escape codes should be generated for function keys. The default is `?false?`, i.e., this feature is disabled.

The `keyboardType` resource is the preferred mechanism for selecting this mode.

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. The default is `?false?`, i.e., this feature is disabled.

The `keyboardType` resource is the preferred mechanism for selecting this mode.

sunKeyboard (class SunKeyboard)

Xterm translates certain key symbols based on its assumptions about your keyboard. This resource specifies whether or not Sun/PC keyboard layout (i.e., the PC keyboard's numeric keypad together with 12 function keys) should be assumed rather than DEC VT220. This causes the keypad `?+?` to be mapped to `?,?` and CTRL F1-F10 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 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 corresponding to the TERM environment variable should be generated for function keys instead of those configured using sunKeyboard and keyboardType. The default is ?false?, i.e., this feature is disabled.

The keyboardType resource is the preferred mechanism for selecting this mode.

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.

Except where noted, they may be bound to characters. Other keywords set modes. Not all keywords are supported on a given system. Allowable keywords include:

Keyword	POSIX?	Notes
---------	--------	-------

??

brk no CHAR may send an ?interrupt? signal, as well as ending the input-line.

dsusp no CHAR will send a terminal ?stop? signal after input is flushed.

eof yes CHAR will terminate input (i.e., an end of file).

eol yes CHAR will end the line.

eol2 no alternate CHAR for ending the line.

erase yes CHAR will erase the last character typed.

erase2 no alternate CHAR for erasing the last input-character.

flush no CHAR will cause output to be discarded until another flush character is typed.

intr yes CHAR will send an ?interrupt? signal.

kill yes CHAR will erase the current line.

lnext no CHAR will enter the next character quoted.

quit yes CHAR will send a ?quit? signal.

rprnt no CHAR will redraw the current line.

start yes CHAR will restart the output after stopping it.

status no CHAR will cause a system-generated status line to be printed.

stop yes CHAR will stop the output.

susp yes CHAR will send a terminal ?stop? signal

swtch no CHAR will switch to a different shell layer.

tabs yes Mode disables tab-expansion.

-tabs yes Mode enables tab-expansion.

weras no CHAR will erase the last word typed.

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 run 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. The POSIX column in the table indicates which keywords are supported by a standard stty program.

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. (This resource is ignored on most systems, because TERMCAP is not used). 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?.

validShells (class ValidShells)

Augment (add to) the system's /etc/shells, when determining whether to set the ?SHELL? environment variable when running a given program.

The resource value is a list of lines (separated by newlines).

Each line holds one pathname. Xterm ignores any line beginning with ?#? after trimming leading/trailing whitespace from each line.

The default is an empty string.

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., using the size according to the window manager. Otherwise, it uses the size as given in resource values or command-line option `-geometry`. 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?`.

zIconTitleFormat (class ZIconTitleFormat)

Allow customization of the string used in the `zIconBeep` feature. The default value is `?*** %s?`.

If the resource value contains a `?%s?`, then xterm inserts the icon title at that point rather than prepending the string to the icon title. (Only the first `?%s?` is used).

VT100 Widget Resources

The following resources are specified as part of the `vt100` widget (class `VT100`). They are specified by patterns such as `?XTerm.vt100.NAME?`.

If your xterm is configured to support the `?toolbar?`, then those patterns need an extra level for the form-widget which holds the toolbar and `vt100` widget. A wildcard between the top-level `?XTerm?` and the `?vt100?` widget makes the resource settings work for either, e.g., `?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 ?default?.

Xterm accepts either a keyword (ignoring case) or the number shown in parentheses:

false (0)

No active icon is shown.

true (1)

The active icon is shown. If you are using twm, use this setting to enable active-icons.

default (2)

Xterm checks at startup, and shows an active icon only for window managers which it can identify and which are known to support the feature. These are fvwm (full support), and window maker (limited). A few other windows managers (such as twm and ctwm) support active icons, but do not support the extensions which allow xterm to identify the window manager.

allowBoldFonts (class AllowBoldFonts)

When set to ?false?, xterm will not use bold fonts. This overrides both the alwaysBoldMode and the boldMode resources.

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?.

allowColorOps (class AllowColorOps)

Specifies whether control sequences that set/query the dynamic colors should be allowed. ANSI colors are unaffected by this resource setting. The default is ?true?.

allowFontOps (class AllowFontOps)

Specifies whether control sequences that set/query the font should be allowed. The default is `?true?`.

`allowMouseOps` (class `AllowMouseOps`)

Specifies whether control sequences that enable xterm to send escape sequences to the host on mouse-clicks and movement. The default is `?true?`.

`allowPasteControls` (class `AllowPasteControls`)

If true, allow control characters such as BEL and CAN to be pasted. Formatting characters (tab, newline) are always allowed. Other C0 control characters are suppressed unless this resource is enabled. The exact set of control characters (C0 and C1) depends upon whether UTF-8 encoding is used, as well as the `allowC1Printable` resource. The default is `?false?`.

`allowScrollLock` (class `AllowScrollLock`)

Specifies whether control sequences that set/query the Scroll Lock key should be allowed, as well as whether the Scroll Lock key responds to user's keypress. The default is `?false?`.

When this feature is enabled, xterm will sense the state of the Scroll Lock key each time it acquires focus. Pressing the Scroll Lock key toggles xterm's internal state, as well as toggling the associated LED. While the Scroll Lock is active, xterm attempts to keep a viewport on the same set of lines. If the current viewport is scrolled past the limit set by the `saveLines` resource, then Scroll Lock has no further effect. The reason for setting the default to `?false?` is to avoid user surprise. This key is generally unused in keyboard configurations, and has not acquired a standard meaning even when it is used in that manner. Consequently, users have assigned it for ad hoc purposes.

`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 would create a very large security hole, therefore enabling this resource forcefully disables the allowXXXOps resources. The default is ?false?.

allowTcapOps (class AllowTcapOps)

Specifies whether control sequences that query the terminal's notion of its function-key strings, as termcap or terminfo capabilities should be allowed. The default is ?true?.

A few programs, e.g., vim, use this feature to get an accurate description of the terminal's capabilities, independent of the termcap/terminfo setting:

? Xterm can tell the querying program how many colors it supports. This is a constant, depending on how it is compiled, typically 16. It does not change if you alter resource settings, e.g., the boldColors resource.

? Xterm can tell the querying program what strings are sent by modified (shift-, control-, alt-) function- and keypad-keys. Reporting control- and alt-modifiers is a feature that relies on the ncurses extended naming.

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. These include several control sequences which manipulate the window size or position, as well as reporting these values and the title or icon name. Each of these can be abused in a script; curiously enough most terminal emulators that implement these restrict only a small part of the repertoire. For fine-tuning, see disallowedWindowOps. 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?`. Xterm provides a menu option for toggling this resource.

`alternateScroll` (class `ScrollCond`)

If `?true?`, the scroll-back and scroll-forw actions send cursor-up and -down keys when xterm is displaying the alternate screen. The default is `?false?`.

The `alternateScroll` state can also be set using a control sequence.

`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

This resource is used only for bitmap fonts:

? When using bitmap fonts, it is possible that the font server will approximate the bold font by rescaling it from a different font size than expected. The alwaysBoldMode resource allows the user to override the (sometimes poor) resulting bold font with overstriking (which is at least consistent).

? The problem does not occur with TrueType fonts (though there can be other unnecessary issues such as different coverage of the normal and bold fonts).

As an alternative, setting the allowBoldFonts resource to false overrides both the alwaysBoldMode and the boldMode resources.

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. Normally xterm checks if Alt or Meta is used in a translation that would conflict with function key modifiers, and will ignore these modifiers in that special case. 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?`.

assumeAllChars (class AssumeAllChars)

If `?true?`, this enables a special case in bitmap fonts to allow the font server to choose how to display missing glyphs. The default is `?true?`.

The reason for this resource is to help with certain quasi-automatically generated fonts (such as the ISO-10646-1 encoding of Terminus) which have incorrect font-metrics.

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 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. A `?true?` value specifies backspace. The default is `?False?`. 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. This font must be the same height and width as the normal font, otherwise it is ignored. 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.

See also the discussion of `boldMode` and `alwaysBoldMode` resources.

`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 font7). 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 introduced a feature which can be used to suppress the scaling. In the X server's configuration file (e.g., `/etc/X11/xorg.conf?`), 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/share/X11/fonts/misc/"
```

would become

```
FontPath      "/usr/share/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?`.

The bitmap scaling feature is also used by xterm to implement VT102 double-width and double-height characters.

`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?.

Xterm's state parser recognizes several types of control strings which can contain text, e.g.,

APC (Application Program Command),

DCS (Device Control String),

OSC (Operating System Command),

PM (Privacy Message), and

SOS (Start of String),

Each should end with a string-terminator (a special character which cannot appear in these strings). Ordinary control characters found within the string are not ignored; they are processed without interfering with the process of accumulating the control string's content. Xterm recognizes these controls in all modes, although some of the functions may be suppressed after parsing the control.

When enabled, this feature allows the user to exit from an unterminated control string when any of these ordinary control characters are found:

control/D (used as an end of file in many shells),

control/H (backspace),

control/I (tab-feed),

control/J (line feed aka newline),
control/K (vertical tab),
control/L (form feed),
control/M (carriage return),
control/N (shift-out),
control/O (shift-in),
control/Q (XOFF),
control/X (cancel)

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 double-sized fonts altogether.

cdXtraScroll (class CdXtraScroll)

Specifies whether xterm should scroll to a new page when clearing the whole screen. Like `tiXtraScroll`, the intent of this option is to provide a picture of the full-screen application's display on the scrollbar before wiping out the text. The default for this resource is `?false?`.

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.

checksumExtension (class ChecksumExtension)

DEC VT420 and up support a control sequence `DECRQCRA` which reports the checksum of the characters in a rectangle. Xterm supports this, with extensions that can be configured with bits of the `checksumExtension`:

- 0 do not negate the result.
- 1 do not report the VT100 video attributes.
- 2 do not omit checksum for blanks.
- 3 omit checksum for cells not explicitly initialized.
- 4 do not mask cell value to 8 bits or ignore combining characters.
- 5 do not mask cell value to 7 bits.

With the default value (0), xterm matches the behavior of DEC's terminals. To use all extensions, set all bits, ?-1? for example.

`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, gray50, red, green, yellow, a customized 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.

Resources past color15 are available as a compile-time option.

Due to a hardcoded limit in the X libraries on the total number of resources (to 400), the resources for 256-colors are omitted

when wide-character support and luit are enabled. Besides

inconsistent behavior if only part of the resources were

allowed, determining the exact cutoff is difficult, and the X

libraries tend to crash if the number of resources exceeds the

limit. The color palette is still initialized to the same

default values, and can be modified via control sequences.

On the other hand, the resource limit does permit including the

entire range for 88-colors.

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?`.

See also the `veryBoldColors` resource which allows combining bold and color.

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?`.

See also the `veryBoldColors` resource which allows combining underline and color.

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?`.

colorIT (class ColorIT)

This specifies the color to use to display italic characters if the `?colorITMode?` resource is enabled. The default is `?XtDefaultForeground?`.

See also the `veryBoldColors` resource which allows combining attributes and color.

colorITMode (class ColorAttrMode)

Specifies whether characters with the italic attribute should be displayed in color or as italic characters. The default is `?false?`.

Note that:

- ? Setting `colorMode` off disables all colors, including italic.
- ? The `italicULMode` resource overrides `colorITMode`.

`colorInnerBorder` (class `ColorInnerBorder`)

Normally, xterm fills the VT100 window's inner border using the background color.

If the `colorInnerBorder` resource is enabled, at startup xterm will compare the `borderColor` and the window's background color.

If those are different, xterm will use the `borderColor` resource to fill the inner border. Otherwise, it will use the window's background color.

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?`.

See also the `veryBoldColors` resource which allows combining reverse and color.

`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?`.

See also the `veryBoldColors` resource which allows combining underline and color.

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 5. 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. Xterm accepts either a keyword (ignoring case) or the number shown in parentheses:

false (0)

The cursor will not blink, but may be combined with escape sequences according to the cursorBlinkXOR resource.

true (1)

The cursor will blink, but may be combined with escape sequences according to the cursorBlinkXOR resource.

always (2)

The cursor will always blink, ignoring escape sequences.

The menu entry will be disabled.

never (3)

The cursor will never blink, ignoring escape sequences. The menu entry will be disabled.

The default is `?false?`.

`cursorBlinkXOR` (class `CursorBlinkXOR`)

Xterm uses two inputs to determine whether the cursor blinks:

? The `cursorBlink` resource (which can be altered with a menu entry).

? Control sequences (private mode 12 and `DECSCUSR`).

The `cursorBlinkXOR` resource determines how those inputs are combined:

`false`

Xterm uses the logical-OR of the two variables. If either is set, xterm makes the cursor blink.

`true`

Xterm uses the logical-XOR of the two variables. If only one is set, xterm makes the cursor blink.

The default is `?true?`.

`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.

cursorUnderLine (class CursorUnderLine)

Specifies whether to make the cursor underlined or a box. The default is false.

cutNewline (class CutNewline)

If false, triple clicking to select a line does not include the newline at the end of the line. If true, the Newline is selected. The default is true.

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.

decGraphicsID (class DecGraphicsID)

Allows a way to combine the graphics feature from certain DEC terminals (125, 240, 241, 330, 340 or 382) with other emulation levels which did not provide the graphics feature. As in decTerminalID, leading non-digit characters are ignored, e.g., vt340 and 340 are the same.

If the resource value is nonzero, xterm uses that emulation level when initializing the drawing region and decoding control sequences to draw graphics.

The default is 0.

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 420.

defaultString (class DefaultString)

Specify the character (or string) which xterm will substitute when pasted text includes a character which cannot be represented in the current encoding. For instance, pasting

UTF-8 text into a display of ISO-8859-1 characters will only be able to display codes 0?255, while UTF-8 text can include Unicode values above 255. The default is `?#?` (a single pound sign).

If the undisplayable text would be double-width, xterm will add a space after the `?#?` character, to give roughly the same layout on the screen as the original text.

`deleteIsDEL` (class `DeleteIsDEL`)

Specifies what the Delete key on the editing keypad should send when pressed. The resource value is a string, evaluated as a boolean after startup. Xterm uses it in conjunction with the `keyboardType` resource:

- ? If the keyboard type is `?default?`, or `?vt220?` and the resource is either `?true?` or `?maybe?` send the VT220-style Remove escape sequence. Otherwise, send DEL (127).
- ? If the keyboard type is `?legacy?`, and the resource is `?true?` send DEL. Otherwise, send the Remove sequence.
- ? Otherwise, if the keyboard type is none of these special cases, send DEL (127).

The default is `?Maybe?`. The resource is allowed to be a non-boolean `?maybe?` so that the popup menu Delete is DEL entry does not override the keyboard type.

`directColor` (class `DirectColor`)

Specifies whether to handle direct-color control sequences using the X server's available colors, or to approximate those using a color map with 256 entries. A `?true?` value enables the former. The default is `?true?`.

`disallowedColorOps` (class `DisallowedColorOps`)

Specify which features will be disabled if `allowColorOps` is false. This is a comma-separated list of names. The default value is

`SetColor,GetColor,GetAnsiColor`

The names are listed below. Xterm ignores capitalization, but

they are shown in mixed-case for clarity.

SetColor

Set a specific dynamic color.

GetColor

Report the current setting of a given dynamic color.

GetAnsiColor

Report the current setting of a given ANSI color (actually any of the colors set via ANSI-style controls).

disallowedFontOps (class DisallowedFontOps)

Specify which features will be disabled if allowFontOps is false. This is a comma-separated list of names. The default value is

SetFont,GetFont

The names are listed below. Xterm ignores capitalization, but they are shown in mixed-case for clarity.

SetFont

Set the specified font.

GetFont

Report the specified font.

disallowedMouseOps (class DisallowedMouseOps)

Specify which features will be disabled if allowMouseOps is false. This is a comma-separated list of names. The default value is `??` which matches all names. The names are listed below. Xterm ignores capitalization, but they are shown in mixed-case for clarity.

X10 The original X10 mouse protocol.

Locator

DEC locator mode

VT200Click

X11 mouse-clicks only.

VT200Hilite

X11 mouse-clicks and highlighting.

AnyButton

XFree86 xterm any-button mode sends button-clicks as well as motion events while the button is pressed.

AnyEvent

XFree86 xterm any-event mode sends button-clicks as well as motion events whether or not a button is pressed.

FocusEvent

Send FocusIn/FocusOut events.

Extended

The first extension beyond X11 mouse protocol, this encodes the coordinates in UTF-8. It is deprecated in favor of SGR, but provided for compatibility.

SGR This is the recommended extension for mouse-coordinates

URXVT

Like Extended, this is provided for compatibility.

AlternateScroll

This overrides the alternateScroll resource.

disallowedPasteControls (class DisallowedPasteControls)

The allowPasteControls resource is normally used to prevent pasting C1 controls, as well as non-formatting C0 controls such as the ASCII escape character. Those characters are simply ignored. This resource further extends the set of control characters which cannot be pasted, converting each into a space.

The resource value is a comma-separated list of names. Xterm ignores capitalization. The default value is

BS,DEL,ENQ,EOT,ESC,NUL

The names are listed below:

C0 all ASCII control characters.

Individual C0 characters

NUL, SOH, STX, ETX, EOT, ENQ, ACK, BEL, BS, HT, LF, VT,
FF, CR, SO, SI, DLE, DC1, DC2, DC3, DC4, NAK, SYN, ETB,
CAN, EM, SUB, ESC, FS, GS, RS, US

DEL ASCII delete

NL ASCII line-feed, i.e., ?newline? is the same as LF.

disallowedTcapOps (class DisallowedTcapOps)

Specify which features will be disabled if allowTcapOps is false. This is a comma-separated list of names. The default value is

SetTcap,GetTcap

The names are listed below. Xterm ignores capitalization, but they are shown in mixed-case for clarity.

SetTcap

(not implemented)

GetTcap

Report specified function- and other special keys.

disallowedWindowOps (class DisallowedWindowOps)

Specify which features will be disabled if allowWindowOps is false. This is a comma-separated list of names, or (for the controls adapted from dtterm the operation number). The default value is

20,21,SetXprop,SetSelection

(i.e., all except a few ?dangerous? operations are allowed).

The names are listed below. Xterm ignores capitalization, but they are shown in mixed-case for clarity. Where a number can be used as an alternative, it is given in parentheses after the name.

GetChecksum

Report checksum of characters in a rectangular region.

GetIconTitle (20)

Report xterm window's icon label as a string.

GetScreenSizeChars (19)

Report the size of the screen in characters as numbers.

GetSelection

Report selection data as a base64 string.

GetWinPosition (13)

Report xterm window position as numbers.

GetWinSizeChars (18)

Report the size of the text area in characters as numbers.

GetWinSizePixels (14)

Report xterm window in pixels as numbers.

GetWinState (11)

Report xterm window state as a number.

GetWinTitle (21)

Report xterm window's title as a string.

LowerWin (6)

Lower the xterm window to the bottom of the stacking order.

MaximizeWin (9)

Maximize window (i.e., resize to screen size).

FullscreenWin (10)

Use full screen (i.e., resize to screen size, without window decorations).

MinimizeWin (2)

Iconify window.

PopTitle (23)

Pop title from internal stack.

PushTitle (22)

Push title to internal stack.

RaiseWin (5)

Raise the xterm window to the front of the stacking order.

RefreshWin (7)

Refresh the xterm window.

RestoreWin (1)

De-iconify window.

SetChecksum

Modify algorithm for reporting checksum of characters in a rectangular region.

SetSelection

Set selection data.

SetWinLines

Resize to a given number of lines, at least 24.

SetWinPosition (3)

Move window to given coordinates.

SetWinSizeChars (8)

Resize the text area to given size in characters.

SetWinSizePixels (4)

Resize the xterm window to given size in pixels.

SetXprop

Set X property on top-level window.

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, modified according to the `eightBitMeta` resource. If `?false?`, Meta characters are converted into a two-character sequence with the character itself preceded by ESC. The default is `?true?`.

The `metaSendsEscape` and `altSendsEscape` resources may override this feature. 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(3x)` function in `curses`.

Note that the Alt key is not necessarily the same as the Meta modifier. The `xmodmap` utility 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.

The `eightBitInput` resource is tested at startup time. If `?true?`, the xterm tries to put the terminal into 8-bit mode.

If `?false?`, on startup, xterm tries to put the terminal into 7-bit mode. For some configurations this is unsuccessful; failure is ignored. After startup, xterm does not change the terminal between 8-bit and 7-bit mode.

As originally implemented in X11, the resource value did not change after startup. However (since patch #216 in 2006) xterm can modify `eightBitInput` after startup via a control sequence.

The corresponding terminfo capabilities `smm` (set meta mode) and `rmm` (reset meta mode) have been recognized by bash for some time. Interestingly enough, bash's notion of `?meta mode?` differs from the standard definition (in the terminfo manual), which describes the change to the eighth bit of a character.

It happens that bash views `?meta mode?` as the ESC character that xterm puts before a character when a special meta key is pressed. bash's early documentation talks about the ESC character and ignores the eighth bit.

`eightBitMeta` (class `EightBitMeta`)

This controls the way xterm modifies the eighth bit of a single-byte key when the `eightBitInput` resource is set. The

default is ?locale?.

The resource value is a string, evaluated as a boolean after startup.

false

The key is sent unmodified.

locale

The key is modified only if the locale uses eight-bit encoding.

true The key is sent modified.

never

The key is always sent unmodified.

Except for the never choice, xterm honors the terminfo capabilities `smm` (set meta mode) and `rmm` (reset meta mode), allowing the feature to be turned on or off dynamically.

If `eightBitMeta` is enabled when the locale uses UTF-8, xterm encodes the value as UTF-8 (since patch #183 in 2003).

`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.

`eightBitSelectTypes` (class `EightBitSelectTypes`)

Override xterm's default selection target list (see `SELECT/PASTE`) for selections in normal (ISO-8859-1) mode. The default is an empty string, i.e., `??`, which does not override anything.

`eraseSavedLines` (class `EraseSavedLines`)

Specifies whether or not to allow xterm extended ED/DECSED control sequences to erase the saved-line buffer. The default is ?true?.

`faceName` (class `FaceName`)

Specify the pattern for scalable fonts selected from the FreeType library if support for that library was compiled into xterm. There is no default value.

One or more fonts can be specified, separated by commas. If prefixed with `?x:?` or `?x11:?` the specification applies to the XLFD font resource. A `?xft:?` prefix is accepted but unnecessary since a missing prefix for `faceName` means that it will be used for TrueType. For example,

```
XTerm*faceName: x:fixed,xft:Bitstream Vera Sans Mono
```

If no `faceName` resource is specified, or if there is no match for both TrueType normal and bold fonts, `xterm` uses the XLFD (bitmap) font and related resources.

It is possible to select suitable bitmap fonts using a script such as this:

```
#!/bin/sh
FONT=`xfontsel -print`
test -n "$FONT" && xfd -fn "$FONT"
```

However (even though `xfd` accepts a `?-fa?` option to denote FreeType fonts), `xfontsel` has not been similarly extended. As a workaround, you may try

```
fc-list :scalable=true:spacing=mono: family
```

to find a list of scalable fixed-pitch fonts which may be used for the `faceName` resource value.

`faceNameDoublesize` (class `FaceNameDoublesize`)

Specify a double-width scalable font for cases where an application requires this, e.g., in CJK applications. There is no default value.

Like the `faceName` resource, this allows one or more comma-separated font specifications to be applied to the wide TrueType or XLFD fonts.

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 8.0. On the VT Fonts menu, this corresponds to the Default entry.

Although the default is 8.0, this may not be the same as the pointsize for the default bitmap font, i.e., that assigned with the -fn option, or the font resource. The default value of faceSize is chosen to match the size of the fixed font, making switching between bitmap and TrueType fonts via the font menu give comparable sizes for the window. If your -fn option uses a different pointsize, you might want to adjust the faceSize resource to match.

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 corresponding bitmap font resources to obtain a TrueType pointsize.

If all of the faceSize resources are set, then xterm will use this information to determine the next smaller/larger TrueType font for the larger-vt-font() and smaller-vt-font() actions.

If any are not set, xterm will use only the areas of the bitmap fonts.

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.

fastScroll (class FastScroll)

Modifies the effect of jump scroll (jumpScroll) by suppressing screen refreshes for the special case when output to the screen has completely shifted the contents off-screen. For instance, cat'ing a large file to the screen does this.

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, corresponding to ?Unreadable? in the standard menu.

font2 (class Font2)

Specifies the name of the second alternative font, corresponding to ?Tiny? in the standard menu.

font3 (class Font3)

Specifies the name of the third alternative font, corresponding to ?Small? in the standard menu.

font4 (class Font4)

Specifies the name of the fourth alternative font, corresponding to ?Medium? in the standard menu.

font5 (class Font5)

Specifies the name of the fifth alternative font, corresponding to ?Large? in the standard menu.

font6 (class Font6)

Specifies the name of the sixth alternative font, corresponding

to ?Huge? in the standard menu.

font7 (class Font7)

Specifies the name of the seventh alternative font, corresponding to ?Enormous? in the standard menu.

fontDoublesize (class FontDoublesize)

Specifies whether xterm should attempt to use font scaling to draw double-sized characters. Some older font servers cannot do this properly, will return misleading font metrics. The default is ?true?. If disabled, xterm will simulate double-sized characters by drawing normal characters with spaces between them.

fontWarnings (class FontWarnings)

Specify whether xterm should report an error if it fails to load a font:

- 0 Never report an error (though the X libraries may).
- 1 Report an error if the font name was given as a resource setting.
- 2 Always report an error on failure to load a font.

The default is ?1?.

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?.

The VT100 line-drawing character set (also known as the DEC Special Character and Line Drawing Set) is shown in this table.

It includes a few special characters which are not used for drawing lines:

Cell	Unicode	Description
0	U+25AE	black vertical rectangle
1	U+25C6	black diamond
2	U+2592	medium shade
3	U+2409	symbol for horizontal tabulation
4	U+240C	symbol for form feed
5	U+240D	symbol for carriage return
6	U+240A	symbol for line feed
7	U+00B0	degree sign
8	U+00B1	plus-minus sign
9	U+2424	symbol for newline
10	U+240B	symbol for vertical tabulation
11	U+2518	box drawings light up and left
12	U+2510	box drawings light down and left
13	U+250C	box drawings light down and right
14	U+2514	box drawings light up and right
15	U+253C	box drawings light vertical and horizontal
16	U+23BA	box drawings scan 1
17	U+23BB	box drawings scan 3
18	U+2500	box drawings light horizontal
19	U+23BC	box drawings scan 7
20	U+23BD	box drawings scan 9
21	U+251C	box drawings light vertical and right
22	U+2524	box drawings light vertical and left
23	U+2534	box drawings light up and horizontal
24	U+252C	box drawings light down and horizontal
25	U+2502	box drawings light vertical
26	U+2264	less-than or equal to

27 U+2265 greater-than or equal to

28 U+03C0 greek small letter pi

29 U+2260 not equal to

30 U+00A3 pound sign

31 U+00B7 middle dot

??

forcePackedFont (class ForcePackedFont)

Specifies whether xterm should use the maximum or minimum glyph width when displaying using a bitmap font. Use the maximum width to help with proportional fonts. The default is ?true?, denoting the minimum width.

forceXftHeight (class ForceXftHeight)

Specifies whether xterm should use the given font metrics for TrueType fonts, or amend the ascent/descent to total no more than the given font-height. This optional feature is used to work around inconsistencies in FreeType's rounding computation. The default is ?false?, denoting the given metrics.

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 VTxxx 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 bg/fg
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)

Tells xterm whether to highlight all of the selected positions,

or only the selected text:

? 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 not 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?`.

inputMethod (class InputMethod)

Tells xterm which type of input method to use. There is no default method.

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?`. See `fastScroll` for a variation.

keepClipboard (class KeepClipboard)

Specifies whether xterm will reuse the selection data which it copied to the clipboard rather than asking the clipboard for its current contents when told to provide the selection. The default is `?false?`.

If compiled into xterm, the menu entry `Keep Clipboard` allows you to change this at runtime.

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 ?true?.

The menu entry Keep Selection allows you to change this at runtime.

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.

limitFontsets (class LimitFontsets)

Limits the number of TrueType fallback fonts (i.e., fontset) which can be used. The default is ?50?.

This limits the number of fallback fonts which xterm uses to display characters. Because TrueType fonts typically are small, xterm may open several fonts for good coverage, and may open additional fonts to obtain information. You can see which font-files xterm opens by setting the environment variable XFT_DEBUG to 3. The Xft library and xterm write this debugging trace to the standard output.

Set this to ?0? to disable fallbacks entirely.

limitResize (class LimitResize)

Limits resizing of the screen via control sequence to a given multiple of the display dimensions. The default is ?1?.

limitResponse (class LimitResponse)

Limits the buffer-size used when xterm replies to various control sequences. The default is ?1024?. The minimum value is ?256?.

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.

For instance, you could have this in your resource file:

```
*VT100.font: 12x24
```

```
*VT100.utf8Fonts.font:9x15
```

When started with a UTF-8 locale, xterm would use 9x15, but allow you to switch to the 12x24 font using the menu entry ?UTF-8 Fonts?.

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. If the encoding converter requires command-line parameters, you can add those after the command, e.g.,

```
*localeFilter: xterm-filter -p
```

Alternatively, you may put those parameter within a shell script to execute the converter, and set this resource to point to the shell script.

When using a locale-filter, e.g., with the `-e` option, or the shell, xterm first tries passing control via that filter. If it fails, xterm will retry without the locale-filter. Xterm warns about the failure before retrying.

logFile (class Logfile)

Specify the name for xterm's log file. If no name is specified, xterm will generate a name when logging is enabled, as described in the `-l` option.

logInhibit (class LogInhibit)

If `?true?`, prevent the logging feature from being enabled, whether by the command-line option `-l`, or the menu entry Log to File. The default is `?false?`.

logging (class Logging)

If ?true?, (and if logInhibit is not set) enable the logging feature. This resource is set/updated by the -l option and the menu entry Log to File. The default is ?false?.

loginShell (class LoginShell)

Specifies whether or not the shell to be run in the window should be started as a login shell. The default is ?false?.

marginBell (class MarginBell)

Specifies whether or not the bell should be rung when the user types near the right margin. The default is ?false?.

maxGraphicSize (class MaxGraphicSize)

If xterm is configured to support ReGIS or SIXEL graphics, this resource controls the maximum size of a graph which can be displayed.

The default is ?1000x1000? (given as width by height).

If the resource is ?auto? then xterm will use the decGraphicsID resource (or decTerminalID if that is not set):

Result decGraphicsID

????????????????????????????

768x400	125
800x460	240
800x460	241
800x480	330
800x480	340
860x750	382
800x480	other

metaSendsEscape (class MetaSendsEscape)

Tells xterm what to do with input-characters modified by Meta:

? 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 ?True?.

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 655 (one percent of the default value for mkSampleSize).

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 65536.

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

To disable mkWidth, and use the system's tables, set both mkSampleSize and mkSamplePass to ?0?. Doing that may make xterm more consistent with applications running in xterm, but may omit some font glyphs whose width correctly differs from the system's character 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?`:

- 1 disables the feature.
- 0 uses the old/obsolete behavior, i.e., the modifier is the first parameter.
- 1 prefixes modified sequences with CSI.
- 2 forces the modifier to be the second parameter if it would otherwise be the first.
- 3 marks 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`:

- 1 permits the user to use shift- and control-modifiers to construct function-key strings using the normal encoding scheme.
- 0 uses the old/obsolete behavior, i.e., the modifier is the first parameter.
- 1 prefixes modified sequences with CSI.
- 2 forces the modifier to be the second parameter if it would otherwise be the first.
- 3 marks 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.

`modifyKeyboard` (class `ModifyKeyboard`)

Normally `xterm` makes a special case regarding modifiers (shift, control, etc.) to handle special keyboard layouts (legacy and `vt220`). This is done to provide compatible keyboards for DEC `VT220` and related terminals that implement user-defined keys (UDK).

The bits of the resource value selectively enable modification of the given category when these keyboards are selected. The default is `?0?`:

- 0 The legacy/`vt220` keyboards interpret only the Control-modifier when constructing numbered function-keys. Other special keys are not modified.
- 1 allows modification of the numeric keypad
- 2 allows modification of the editing keypad
- 4 allows modification of function-keys, overrides use of Shift-modifier for UDK.
- 8 allows modification of other special keys

`modifyOtherKeys` (class `ModifyOtherKeys`)

Like `modifyCursorKeys`, tells `xterm` to construct an escape sequence for ordinary (i.e., `?other?`) keys (such as `?2?`) when modified by Shift-, Control-, Alt- or Meta-modifiers. This feature does not apply to special keys, i.e., cursor-, keypad-, function- or control-keys which are labeled on your keyboard.

Those have key symbols which XKB identifies uniquely.

For example, this feature does not apply to special control-keys (e.g., Escape, Tab, Enter, Backspace) Other control keys (e.g., Control-I, Control-M, Control-H) may send escape sequences when this feature is enabled.

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 which are built into the X11 library, e.g., Control-Space to make a NUL, or Control-3 to make an Escape character.

Except for those special cases built into the X11 library, the Shift- and Control- modifiers are treated normally.

The Alt- and Meta- modifiers do not cause xterm to send escape sequences. Those modifier keys are interpreted according to other resources, e.g., the metaSendsEscape resource.

- 2 enables this feature for keys including the exceptions listed. Xterm ignores the special cases built into the X11 library. Any shifted (modified) ordinary key sends an escape sequence. The Alt- and Meta- modifiers cause xterm to send escape sequences.

The Xterm FAQ has an extended discussion of this feature, with examples:

<https://invisible-island.net/xterm/modified-keys.html>

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?.

nameKeymap (class NameKeymap)

See the discussion of the keymap() action.

nextEventDelay (class NextEventDelay)

Specifies a delay time in milliseconds before checking for new X events. The default is ?1?.

numColorRegisters (class NumColorRegisters)

If xterm is configured to support ReGIS or SIXEL graphics, this specifies the number of color-registers which are available.

If this resource is not specified, xterm uses a value determined by the decTerminalID resource:

Result decTerminalID

????????????????????????

4	125
4	240
4	241
4	330
16	340
2	382
1024	other

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 (X11R5) escape sequences for function keys F1 to F4, for compatibility with X Consortium xterm. Otherwise, it uses the VT100 codes for PF1 to PF4. The default is ?false?.

Setting this resource has the same effect as setting the keyboardType to legacy. The keyboardType resource is the preferred mechanism for selecting this mode.

The old-style escape sequences resemble VT220 keys, but appear to have been invented for xterm in X11R4.

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 Functions 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.

If the pointer is on a ?word? then xterm searches back to the beginning of the word, and then to the end.

If the pointer is not on a ?word? then the result depends on whether it is on whitespace (including a newline), or past the end of the line. In the latter case xterm may select a ?word? beginning after the newline, if there is no additional whitespace.

line

Select a line (counting wrapping).

group

Select a group of adjacent lines (counting wrapping). The selection stops on a blank line, and does not extend outside the current page.

page

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

all

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

regex

Select the best match for the POSIX extended regular expression (ERE) which follows in the resource value:

? Xterm matches the regular expression against a byte

array for the entire (possibly wrapped) line. That byte array may be UTF-8 or ISO-8859-1, depending on the mode in which xterm is running.

? Xterm steps through each byte-offset in this array, keeping track of the best (longest) match. If more than one match ties for the longest length, the first is used.

Xterm does this to make it convenient to click anywhere in the area of interest and cause the regular expression to match the entire word, etc.

? The `^` and `$` anchors in a regular expression denote the ends of the entire line.

? If the regular expression contains backslashes `\` those should be escaped `\\` because the X libraries interpret backslashes in resource strings.

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 `onXClicks` resource values which are set.

`openIm` (class `OpenIm`)

Tells xterm whether to open the input method at startup. The default is `?true?`.

`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?.

pointerFont (class PointerFont)

Specifies the font to be used for the pointer. The shapes specified by pointerShape are glyphs in this font. The resource value default is cursor.

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

1 the application running in xterm has not activated mouse mode. This is the default.

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?.

If the window is iconified, this has no effect. However, the zIconBeep resource provides you with the ability to see which iconified windows have sounded a bell.

precompose (class Precompose)

Tells xterm whether to precompose UTF-8 data into Normalization Form C, which combines commonly-used accents onto base characters. If it does not do this, accents are left as separate characters. The default is ?true?.

preeditType (class PreeditType)

Tells xterm which types of preedit (preconversion) string to display. The default is ?OverTheSpot,Root?.

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.

? ?0? disables the attributes.

? ?1? prints the normal set of attributes (bold, underline, inverse and blink) as VT100-style control sequences.

? ?2? prints ANSI color attributes as well.

The default is ?1?.

printFileImmediate (class PrintFileImmediate)

When the print-immediate action is invoked, xterm prints the screen contents directly to a file. Set this resource to the prefix of the filename (a timestamp will be appended to the actual name).

The default is an empty string, i.e., ??, However, when the print-immediate action is invoked, if the string is empty, then ?XTerm? is used.

printFileOnError (class PrintFileOnError)

If xterm exits with an X error, e.g., your connection is broken when the server crashes, it can be told to write the contents of the screen to a file. To enable the feature, set this resource to the prefix of the filename (a timestamp will be appended to the actual name).

The default is an empty string, i.e., ??, which disables this feature. However, when the print-on-error action is invoked, if the string is empty, then ?XTermError? is used.

These error codes are handled: ERROR_XERROR, ERROR_XIOERROR and ERROR_ICEERROR.

printModelImmediate (class PrintModelImmediate)

When the print-immediate action is invoked, xterm prints the screen contents directly to a file. You can use the printModelImmediate resource to tell it to use escape sequences to reconstruct the video attributes and colors. This uses the same values as the printAttributes resource. The default is ?0?.

printModeOnError (class PrintModeOnError)

Xterm implements the `printFileOnXError` feature using the printer feature, although the output is written directly to a file. You can use the `printModeOnXError` resource to tell it to use escape sequences to reconstruct the video attributes and colors. This uses the same values as the `printAttributes` resource. The default is `?0?`.

`printOptsImmediate` (class `PrintOptsImmediate`)

Specify the range of text which is printed to a file when the `print-immediate` action is invoked.

? If zero (0), then this selects the current (visible screen) plus the saved lines, except if the alternate screen is being used. In that case, only the alternate screen is selected.

? If nonzero, the bits of this resource value (checked in descending order) select the range:

8 selects the saved lines.

4 selects the alternate screen.

2 selects the normal screen.

1 selects the current screen, which can be either the normal or alternate screen.

The default is `?9?`, which selects the current visible screen plus saved lines, with no special case for the alternated screen.

`printOptsOnXError` (class `PrintOptsOnXError`)

Specify the range of text which is printed to a file when the `print-on-error` action is invoked. The resource value is interpreted the same as in `printOptsImmediate`.

The default is `?9?`, which selects the current visible screen plus saved lines, with no special case for the alternated screen.

`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 an empty string, i.e., `??`. If the resource value is given as an empty string, the printer is disabled.

`printerControlMode` (class `PrinterControlMode`)

Specifies the printer control mode. A `?1?` selects autoprnt 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.

Autoprnt 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 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?`.

`printerNewLine` (class `PrinterNewLine`)

Controls whether a newline is sent to the printer at the end of a print page function. The default is `?true?`.

`privateColorRegisters` (class `PrivateColorRegisters`)

If xterm is configured to support ReGIS or SIXEL graphics, this controls whether xterm allocates separate color registers for each sixel device control string, e.g., for DECGCI. If not true, color registers are allocated only once, when the terminal is reset, and color changes in any graphic affect all graphics. The default is `?true?`.

`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?`.

regisDefaultFont (class RegisDefaultFont)

If xterm is configured to support ReGIS graphics, this resource tells xterm which font to use if the ReGIS data does not specify one. No default value is specified; xterm accepts a TrueType font specification as in the `faceName` resource.

If no value is specified, xterm draws a bitmap indicating a missing character.

regisScreenSize (class RegisScreenSize)

If xterm is configured to support ReGIS graphics, this resource tells xterm the default size (in pixels) for these graphics, which also sets the default coordinate space to `[0,0]` (upper-left) and `[width,height]` (lower-right).

The application using ReGIS may use the `?A?` option of the `?S?` command to adjust the coordinate space or change the addressable portion of the screen.

Xterm accepts a special resource value `?auto?`, which tells xterm to use the `decGraphicsID` and `decTerminalID` resources to set the default size based on the hardware terminal's limits.

Those limits are the same as for the `maxGraphicSize` resource.

The default is `?auto?`.

renderFont (class RenderFont)

If xterm is built with the Xft library, this controls whether the `faceName` resource is used. The default is `?default?`.

The resource values are strings, evaluated as booleans after startup.

false

disable the feature and use the normal (bitmap) font.

true

startup using the TrueType font specified by the `faceName` and `faceSize` resource settings. If there is no value for

faceName, disable the feature and use the normal (bitmap) font.

After startup, you can still switch to/from the bitmap font using the ?TrueType Fonts? menu entry.

default

Enable the ?TrueType Fonts? menu entry to allow runtime switching to/from TrueType fonts. The initial font used depends upon whether the faceName resource is set:

? If the faceName resource is not set, start by using the normal (bitmap) font. Xterm has a separate compiled-in value for faceName for this special case. That is normally ?mono?.

? If the faceName resource is set, then start by using the TrueType font rather than the bitmap font.

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 X11R4. 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.

retryInputMethod (class RetryInputMethod)

Tells xterm how many times to retry, in case the input-method server is not responding. This is a different issue than unsupported preedit type, etc. You may encounter retries if your X configuration (and its libraries) are missing pieces. Setting this resource to zero ?0? will cancel the retrying.

The default is ?3?.

reverseVideo (class ReverseVideo)

Specifies whether or not reverse video should be simulated.

The default is `?false?`.

There are several aspects to reverse video in xterm:

- ? The command-line `-rv` option tells the X libraries to reverse the foreground and background colors. Xterm's command-line options set resource values. In particular, the X Toolkit sets the `reverseVideo` resource when the `-rv` option is used.
- ? If the user has also used command-line options `-fg` or `-bg` to set the foreground and background colors, xterm does not see these options directly. Instead, it examines the resource values to reconstruct the command-line options, and determine which of the colors is the user's intended foreground, etc. Their actual values are irrelevant to the reverse video function; some users prefer the X defaults (black text on a white background), others prefer white text on a black background.
- ? After startup, the user can toggle the `?Enable Reverse Video?` menu entry. This exchanges the current foreground and background colors of the VT100 widget, and repaints the screen. Because of the X resource hierarchy, the `reverseVideo` resource applies to more than the VT100 widget.

Programs running in an xterm can also use control sequences to enable the VT100 reverse video mode. These are independent of the `reverseVideo` resource and the menu entry. Xterm exchanges the current foreground and background colors when drawing text affected by these control sequences.

Other control sequences can alter the foreground and background colors which are used:

- ? Programs can also use the ANSI color control sequences to set the foreground and background colors.

- ? Extensions to the ANSI color controls (such as 16-, 88- or 256-colors) are treated similarly to the ANSI control.
- ? Using other control sequences (the ?dynamic colors? feature), a program can change the foreground and background colors.

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 ?1024?.

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.

`shiftEscape` (class `ShiftEscape`)

Xterm uses the translations resource to determine how to invoke actions for selecting and copying text using the pointer (e.g., a mouse). It also provides a mouse protocol which can be used by applications running in an xterm to detect mouse button clicks.

The mouse protocol causes xterm to send special escape sequences which allow an application to determine if modifiers (i.e., one or more of shift, control, alt, and meta) were used.

Xterm provides this mouse protocol by interpreting button- and motion-events in the functions which the translations resource calls for selecting and copying text:

- insert-selection
- select-end
- select-extend
- select-start
- start-extend

While the mouse protocol is active, xterm reserves most of the mouse button events for sending special escape sequences to the application. Xterm normally allows you to use the shift-key to temporarily override this mouse protocol, permitting the selection and copying actions to be used.

The `shiftEscape` resource setting allows you to tell xterm whether to use the shift-key in this way (i.e., overriding the

mouse protocol). Xterm accepts either a keyword (ignoring case) or the number shown in parentheses:

false (0)

Mouse protocol does not send special escapes when shift-key is used.

true (1)

Mouse protocol may send special escapes when shift-key is used.

At startup, xterm analyzes the translations to see which buttons are used in the (mouse) button-related bindings for selection and copying text. If the shift-key is not mentioned explicitly in a button's binding, xterm allows that button with shift-key for overriding the mouse protocol.

always (2)

Mouse protocol can always send special escapes when shift-key is used.

never (3)

Mouse protocol will never send special escapes when shift-key is used.

Xterm interprets a control sequence which can change this setting between ?true? and ?false?. The default is ?false?.

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?`.

`showWrapMarks` (class `ShowWrapMarks`)

For debugging xterm and applications that may manipulate the wrapped-line flag by writing text at the right margin, show a mark on the right inner-border of the window. The mark shows which lines have the flag set.

`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?`.

`sixelScrolling` (class `SixelScrolling`)

If xterm is configured to support SIXEL graphics, this resource tells it whether to scroll up one line at a time when sixels would be written past the bottom line on the window. The default is `?false?`.

`sixelScrollsRight` (class `SixelScrollsRight`)

If xterm is configured to support SIXEL graphics, this resource tells it whether to scroll to the right as needed to keep the current position visible rather than truncate the plot on the on the right. 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 titelInhibit is ?true?, because the intent of this option is to provide a picture of the full-screen application's display on the scrollbar without wiping out the text that would be shown before the application was initialized. The default for this resource is ?false?.

titelInhibit (class TitelInhibit)

Originally specified 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.

TERMCAP is used rarely now, but xterm supports the feature on modern systems:

- ? 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?.

titleModes (class TitleModes)

Tells xterm whether to accept or return window- and icon-labels in ISO-8859-1 (the default) or UTF-8. Either can be encoded in hexadecimal:

- ? UTF-8 titles require special treatment, because they may contain bytes which can be mistaken for control characters. Hexadecimal-encoding is supported to eliminate that possibility.

- ? As an alternative, you could use the allowC1Printable

resource, which suppresses xterm's parsing of the relevant control characters (and as a result, treats those bytes as data).

The default for this resource is `?0?`.

Each bit (bit `?0?` is 1, bit `?1?` is 2, etc.) corresponds to one of the parameters set by the title modes control sequence:

- 0 Set window/icon labels using hexadecimal
- 1 Query window/icon labels using hexadecimal
- 2 Set window/icon labels using UTF-8 (gives the same effect as the `utf8Title` resource).
- 3 Query window/icon labels using UTF-8

`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?`.

`useBorderClipping` (class `UseBorderClipping`)

Tell xterm whether to apply clipping when useClipping is false.

Unlike useClipping, this simply limits text to keep it within the window borders, e.g., as a refinement to the scaleHeight workaround. The default is ?false?.

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 can be set via the menu entry ?UTF-8 Encoding?. The default is ?default?.

Xterm accepts either a keyword (ignoring case) or the number shown in parentheses:

false (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.

true (1)

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

always (2)

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

default (3)

This is the default value of the resource. It is changed during initialization depending on whether the locale resource was set, to false (0) or always (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. This specifies whether xterm will use UTF-8 fonts specified via resource patterns such as `*vt100.utf8Fonts.font?` or normal (ISO-8859-1) fonts via patterns such as `*vt100.font?`. The resource can be set via the menu entry `?UTF-8 Fonts?`. The default is `?default?`.

Xterm accepts either a keyword (ignoring case) or the number shown in parentheses:

false (0)

Use the ISO-8859-1 fonts. The menu entry is enabled, allowing the choice of fonts to be changed at runtime.

true (1)

Use the UTF-8 fonts. The menu entry is enabled, allowing the choice of fonts to be changed at runtime.

always (2)

Always use the UTF-8 fonts. This also disables the menu entry.

default (3)

At startup, the resource is set to true or false, according to the effective value of the `utf8` resource.

utf8Latin1 (class Utf8Latin1)

If true, allow an ISO-8859-1 normal font to be combined with an ISO-10646-1 font if the latter is given via the `-fw` option or its corresponding resource value. The default is `?false?`.

utf8SelectTypes (class Utf8SelectTypes)

Override xterm's default selection target list (see `SELECT/PASTE`) for selections in wide-character (UTF-8) mode. The default is an empty string, i.e., `??`, which does not override anything.

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. The window manager is responsible for drawing window titles. Some window managers (not all) support UTF-8 encoding of window titles. Set this resource to `?true?` to also set UTF-8 encoded title strings using the EWMH properties.

This feature is available as a menu entry, since it is related to the particular applications you are running within xterm.

You can also use a control sequence (see the discussion of `?Title Modes?` in Xterm Control Sequences), to set an equivalent flag (which can also be set using the `titleModes` resource).

Xterm accepts either a keyword (ignoring case) or the number shown in parentheses:

false (0)

Set only ISO-8859-1 title strings, e.g., using the ICCCM `WM_NAME STRING` property. The menu entry is enabled, allowing the choice of title-strings to be changed at runtime.

true (1)

Set both the EWMH (UTF-8 strings) and the ICCCM `WM_NAME`, etc. The menu entry is enabled, allowing the choice to be changed at runtime.

always (2)

Always set both the EWMH (UTF-8 strings) and the ICCCM `WM_NAME`, etc. This also disables the menu entry.

default (3)

At startup, the resource is set to true or false,

according to the effective value of the utf8 resource.

The default is ?default?.

utf8Weblike (class Utf8Weblike)

Provide an alternate error-handling scheme for ill-formed UTF-8 as recommended in a W3C document. The Unicode standard does not require this for conformance. Some additional information can be found here:

<https://invisible-island.net/xterm/bad-utf8/>

The default is ?false?.

veryBoldColors (class VeryBoldColors)

Specifies whether to combine video attributes with colors specified by colorBD, colorBL, colorIT, colorRV, and colorUL.

The resource value is the sum of values for each attribute:

1 for reverse,

2 for underline,

4 for bold,

8 for blink, and

512 for italic

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?, which tells xterm to use an audible bell.

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.

visualBellLine (class VisualBellLine)

Specifies whether to flash only the current line when displaying a visual bell rather than flashing the entire

screen: The default is ?false?, which tells xterm to flash the

entire screen.

vt100Graphics (class VT100Graphics)

This specifies whether xterm will interpret VT100 graphic character escape sequences while in UTF-8 mode. This feature also applies to code-pages (e.g., for VT320 and VT520) and National Replacement Character Sets (VT220 and up), but not US-ASCII (the initially selected character set), to avoid conflict with UTF-8. 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 double-width 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 double-width 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 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 ?lineN? where N is a number are separators with class SmeLine.

As with all X resource-based widgets, the labels mentioned are customary defaults for the application.

The Main Options menu (widget name 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-immediate (class SmeBSB)

This entry invokes the print-immediate() action.

print-on-error (class SmeBSB)

This entry invokes the print-on-error() action.

print (class SmeBSB)

This entry invokes the print() action.

print-redir (class SmeBSB)

This entry invokes the print-redir() action.

dump-html (class SmeBSB)

This entry invokes the dump-html() action.

dump-svg (class SmeBSB)

This entry invokes the dump-svg() 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 set-old-function-keys(toggle) action.

hpFunctionKeys (class SmeBSB)

This entry invokes the set-hp-function-keys(toggle) action.

scoFunctionKeys (class SmeBSB)

This entry invokes the set-sco-function-keys(toggle) action.

sunFunctionKeys (class SmeBSB)

This entry invokes the set-sun-function-keys(toggle) action.

sunKeyboard (class SmeBSB)

This entry invokes the sunKeyboard(toggle) action.

suspend (class SmeBSB)

This entry invokes the send-signal(tsfp) 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 VT Options menu (widget name 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.

keepSelection (class SmeBSB)

This entry invokes the set-keep-selection(toggle) action.

selectToClipboard (class SmeBSB)

This entry invokes the set-keep-clipboard(toggle) action.

visualbell (class SmeBSB)

This entry invokes the set-visual-bell(toggle) action.

bellIsUrgent (class SmeBSB)

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

poponbell (class SmeBSB)

This entry invokes the set-pop-on-bell(toggle) action.

cursorblink (class SmeBSB)

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

titelnhibit (class SmeBSB)

This entry invokes the set-titelnhibit(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.

sixelScrolling (class SmeBSB)

This entry invokes the set-sixel-scrolling(toggle) action.

privateColorRegisters (class SmeBSB)

This entry invokes the set-private-colors(toggle) action.

The VT Fonts menu (widget name fontMenu) has the following entries:

fontdefault (class SmeBSB)

This entry invokes the set-vt-font(d) action, setting the font using the font (default) resource, e.g., ?Default? in the menu.

font1 (class SmeBSB)

This entry invokes the set-vt-font(1) action, setting the font using the font1 resource, e.g., ?Unreadable? in the menu.

font2 (class SmeBSB)

This entry invokes the set-vt-font(2) action, setting the font using the font2 resource, e.g., ?Tiny? in the menu.

font3 (class SmeBSB)

This entry invokes the set-vt-font(3) action, setting the font using the font3 resource, e.g., ?Small? in the menu.

font4 (class SmeBSB)

This entry invokes the set-vt-font(4) action, letting the font using the font4 resource, e.g., ?Medium? in the menu.

font5 (class SmeBSB)

This entry invokes the set-vt-font(5) action, letting the font using the font5 resource, e.g., ?Large? in the menu.

font6 (class SmeBSB)

This entry invokes the set-vt-font(6) action, letting the font using the font6 resource, e.g., ?Huge? in the menu.

font7 (class SmeBSB)

This entry invokes the set-vt-font(7) action, letting the font using the font7 resource, e.g., ?Enormous? in the menu.

fontescape (class SmeBSB)

This entry invokes the set-vt-font(e) action.

fontsel (class SmeBSB)

This entry invokes the set-vt-font(s) action.

allow-bold-fonts (class SmeBSB)

This entry invokes the allow-bold-fonts(toggle) action.

font-linedrawing (class SmeBSB)

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

font-packed (class SmeBSB)

This entry invokes the set-font-packed(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-fonts (class SmeBSB)

This entry invokes the set-utf8-fonts(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.

allow-color-ops (class SmeBSB)

This entry invokes the allow-color-ops(toggle) action.

allow-font-ops (class SmeBSB)

This entry invokes the allow-fonts-ops(toggle) action.

allow-tcap-ops (class SmeBSB)

This entry invokes the allow-tcap-ops(toggle) action.

allow-title-ops (class SmeBSB)

This entry invokes the allow-title-ops(toggle) action.

allow-window-ops (class SmeBSB)

This entry invokes the allow-window-ops(toggle) action.

The Tek Options menu (widget name 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:

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.

thickness (class Thickness)

Specifies the width in pixels of the scrollbar (default: 14).

This may be overridden by the width resource.

thumb (class Thumb)

The default ?thumb? pixmap used for the scrollbar is a simple checkerboard pattern alternating pixels for foreground and background color.

width (class Width)

Specifies the width in pixels of the scrollbar (default: 0).

The widget checks the width resource first, using the thickness value if the width is zero.

POINTER USAGE

Once the VTxxx window is created, xterm allows you to select text and copy it within the same or other windows using the pointer or the keyboard.

A ?pointer? could be a mouse, touchpad or similar device. X applications generally do not care, since they see only button events which have

? position and

? button up/down state

Xterm can see these events as long as it has focus.

The keyboard also supplies events, but it is less flexible than the pointer for selecting/copying text.

Events are applied to actions using the translations resource. See Actions for a complete list, and Default Key Bindings for the built-in set of translations resources.

Selection Functions

By default, the selection functions are invoked when the pointer buttons are used with no modifiers, and when they are used with the `?shift?` key. The `?shift?` key is special, because xterm uses that to ensure that selection functions are still available when it is programmed to send escape sequences in one of the mouse modes (see Xterm Control Sequences, as well as the resource `disallowedMouseOps`). At startup, xterm inspects the translations resource to see which pointer buttons may be used in this way, and remembers these buttons when deciding whether to send escape sequences or perform selection when those buttons are used with the `?shift?` modifier. Other pointer buttons, e.g., typically those sent for wheel mouse events, are not affected.

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:

```
~Meta <Btn1Down>:select-start()
```

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 selection when the button is released:

```
<BtnUp>:select-end(SELECT, CUT_BUFFER0) \n
```

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 given selection, if any, otherwise from the cut buffer, inserting it as keyboard input:

```
~Ctrl ~Meta <Btn2Up>:insert-selection(SELECT, CUT_BUFFER0)
```

Pointer button three (usually right)

extends the current selection.

```
~Ctrl ~Meta <Btn3Down>:start-extend()
```

(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 VTxxx 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 bit is normally stripped unless the terminal mode is RAW; see `tty(4)` for details).

SELECT/PASTE

X clients provide select and paste support by responding to requests conveyed by the X server. The X server holds data in ?atoms? which correspond to the different types of selection (PRIMARY, SECONDARY, CLIPBOARD) as well as the similar cut buffer mechanism (CUT_BUFFER0 to CUT_BUFFER7). Those are documented in the ICCCM.

The ICCCM deals with the underlying mechanism for select/paste. It does not mention highlighting. The selection is not the same as highlighting. Xterm (like many applications) uses highlighting to show you the currently selected text. An X application may own a selection, which allows it to be the source of data copied using a given selection

atom Xterm may continue owning a selection after it stops highlighting (see keepSelection).

PRIMARY

When configured to use the primary selection (the default), xterm can provide the selection data in ways which help to retain character encoding information as it is pasted.

The PRIMARY token is a standard X feature, documented in the ICCCM (Inter-Client Communication Conventions Manual), which states

The selection named by the atom PRIMARY is used for all commands that take only a single argument and is the principal means of communication between clients that use the selection mechanism.

A user ?selects? text on xterm, which highlights the selected text. A subsequent ?paste? to another client forwards a request to the client owning the selection. If xterm owns the primary selection, it makes the data available in the form of one or more ?selection targets?. If it does not own the primary selection, e.g., if it has released it or another client has asserted ownership, it relies on cut-buffers to pass the data. But cut-buffers handle only ISO-8859-1 data (officially - some clients ignore the rules).

CLIPBOARD

When configured to use the clipboard (using the selectToClipboard resource), the problem with persistence of ownership is bypassed. Otherwise, there is no difference regarding the data which can be passed via selection.

The selectToClipboard resource is a compromise, allowing CLIPBOARD to be treated almost like PRIMARY, unlike the ICCCM, which describes CLIPBOARD in different terms than PRIMARY or SECONDARY. Its lengthy explanation begins with the essential points:

The selection named by the atom CLIPBOARD is used to hold data that is being transferred between clients, that is, data that usually is being cut and then pasted or copied and then pasted. Whenever a client wants to transfer data to the clipboard:

? It should assert ownership of the CLIPBOARD.

? If it succeeds in acquiring ownership, it should be prepared to respond to a request for the contents of the CLIPBOARD in the usual way (retaining the data to be able to return it). The request may be generated by the clipboard client described below.

SELECT

However, many applications use CLIPBOARD in imitation of other windowing systems. The selectToClipboard resource (and corresponding menu entry Select to Clipboard) introduce the SELECT token (known only to xterm) which chooses between the PRIMARY and CLIPBOARD tokens. Without using this feature, one can use workarounds such as the xclip program to show the contents of the X clipboard within an xterm window.

SECONDARY

This is used less often than PRIMARY or CLIPBOARD. According to the ICCCM, it is used

- ? As the second argument to commands taking two arguments (for example, ?exchange primary and secondary selections?)
- ? As a means of obtaining data when there is a primary selection and the user does not want to disturb it

Selection Targets

The different types of data which are passed depend on what the receiving client asks for. These are termed selection targets.

When asking for the selection data, xterm tries the following types in this order:

UTF8_STRING

This is an XFree86 extension, which denotes that the data is encoded in UTF-8. When xterm is built with wide-character support, it both accepts and provides this type.

TEXT the text is in the encoding which corresponds to your current locale.

COMPOUND_TEXT

this is a format for multiple character set data, such as multi-lingual text. It can store UTF-8 data as a special

case.

STRING

This is Latin 1 (ISO-8859-1) data.

The middle two (TEXT and COMPOUND_TEXT) are added if xterm is configured with the `i18nSelections` resource set to `?true?`.

UTF8_STRING is preferred (therefore first in the list) since xterm stores text as Unicode data when running in wide-character mode, and no translation is needed. On the other hand, TEXT and COMPOUND_TEXT may require translation. If the translation is incomplete, they will insert X's `?defaultString?` whose value cannot be set, and may simply be empty. Xterm's `defaultString` resource specifies the string to use for incomplete translations of the UTF8_STRING.

You can alter the types which xterm tries using the `eightBitSelectTypes` or `utf8SelectTypes` resources. For instance, you might have some specific locale setting which does not use UTF-8 encoding. The resource value is a comma-separated list of the selection targets, which consist of the names shown. You can use the special name `I18N` to denote the optional inclusion of TEXT and COMPOUND_TEXT. The names are matched ignoring case, and can be abbreviated. The default list can be expressed in several ways, e.g.,

```
UTF8_STRING,I18N,STRING
```

```
utf8,i18n,string
```

```
u,i,s
```

Mouse Protocol

Applications can send escape sequences to xterm to cause it to send escape sequences back to the computer when you press a pointer button, or even (depending on which escape sequence) send escape sequences back to the computer as you move the pointer.

These escape sequences and the responses, called the mouse protocol, are documented in XTerm Control Sequences. They do not appear in the actions invoked by the `translations` resource because the resource does not change while you run xterm, whereas applications can change the mouse protocol (i.e., enable, disable, use different modes).

However, the mouse protocol is interpreted within the actions that are usually associated with the pointer buttons. Xterm ignores the mouse protocol in the insert-selection action if the shift-key is pressed at the same time. It also modifies a few other actions if the shift-key is pressed, e.g., suppressing the response with the pointer position, though not eliminating changes to the selected text.

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 VTxxx and Tektronix windows. There are several sections:

Commands for managing X events:

Toolbar (resource toolbar)

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

Secure Keyboard (resource 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 (resource 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 (resource redraw)

Forces the X display to repaint; useful in some environments.

Commands for capturing output:

Log to File (resource logging)

Captures text sent to the screen in a log file, as in the -l logging option.

Print-All Immediately (resource print-immediate)

Invokes the print-immediate action, sending the text of the current window directly to a file, as specified by the printFileImmediate, printModelImmediate and printOptsImmediate resources.

Print-All on Error (resource print-on-error)

Invokes the print-on-error action, which toggles a flag telling xterm that if it exits with an X error, to send the text of the current window directly to a file, as specified by the printFileOnError, printModeOnError and printOptsOnError resources.

Print Window (resource print)

Sends the text of the current window to the program given in the printerCommand resource.

Redirect to Printer (resource 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.

XHTML Screen Dump (resource dump-html)

Available only when compiled with screen dump support. Invokes the dump-html action. This creates an XHTML file matching the contents of the current screen, including the border, internal border, colors and most attributes:

bold, italic, underline, faint, ~~strikeout~~, reverse; blink is rendered as white-on-red; double underline is rendered the same as underline since there is no portable equivalent in CSS 2.2.

The font is whatever your browser uses for preformatted (<pre>) elements. The XHTML file references a cascading style sheet (CSS) named `?xterm.css?` that you can create to select a font or override properties.

The following CSS selectors are used with the expected default behavior in the XHTML file:

`.ul` for underline,

`.bd` for bold,

`.it` for italic,

`.st` for ~~strikeout~~,

`.lu` for ~~strikeout~~ combined with underline.

In addition you may use

`.ev` to affect even numbered lines and

`.od` to affect odd numbered lines.

Attributes faint, reverse and blink are implemented as style attributes setting color properties. All colors are specified as RGB percentages in order to support displays with 10 bits per RGB.

The name of the file will be

```
xterm.yyyy.MM.dd.hh.mm.ss.xhtml
```

where yyyy, MM, dd, hh, mm and ss are the year, month, day, hour, minute and second when the screen dump was performed (the file is created in the directory xterm is started in, or the home directory for a login xterm).

The dump-html action can also be triggered using the Media Copy control sequence CSI 1 0 i, for example from a shell script with

```
printf '\033[10i'
```

Only the UTF-8 encoding is supported.

SVG Screen Dump (resource dump-svg)

Available only when compiled with screen dump support.

Invokes the dump-svg action. This creates a Scalable Vector Graphics (SVG) file matching the contents of the current screen, including the border, internal border, colors and most attributes: bold, italic, underline, double underline, faint, strikeout, reverse; blink is rendered as white-on-red. The font is whatever your renderer uses for the monospace font-family. All colors are specified as RGB percentages in order to support displays with 10 bits per RGB.

The name of the file will be

```
xterm.yyyy.MM.dd.hh.mm.ss.svg
```

where yyyy, MM, dd, hh, mm and ss are the year, month, day, hour, minute and second when the screen dump was performed (the file is created in the directory xterm is started in, or the home directory for a login xterm).

The dump-svg action can also be triggered using the Media Copy control sequence CSI 1 1 i, for example from a shell script with

```
printf '\033[11i'
```

Only the UTF-8 encoding is supported.

Modes for setting keyboard style:

8-Bit Controls (resource 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 Xterm Control Sequences).

This corresponds to the eightBitControl resource.

Backarrow Key (BS/DEL) (resource 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 (resource num-lock)

Controls the treatment of Alt- and NumLock-key modifiers.

This corresponds to the numLock resource.

Meta Sends Escape (resource 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 (resource 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 deletIsDEL resource.

Old Function-Keys (resource oldFunctionKeys)

HP Function-Keys (resource hpFunctionKeys)

SCO Function-Keys (resource scoFunctionKeys)

Sun Function-Keys (resource sunFunctionKeys)

VT220 Keyboard (resource sunKeyboard)

These act as a radio-button, selecting one style for the keyboard layout. The layout corresponds to more than one resource setting: sunKeyboard, sunFunctionKeys, scoFunctionKeys and hpFunctionKeys.

Commands for process signalling:

Send STOP Signal (resource suspend)

Send CONT Signal (resource continue)

Send INT Signal (resource interrupt)

Send HUP Signal (resource hangup)

Send TERM Signal (resource terminate)

Send KILL Signal (resource 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 (resource quit)

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

VT Options

The xterm vtMenu sets various modes in the VTxxx emulation, and is popped up when the ?control? key and pointer button two are pressed in the VTxxx window.

VTxxx Modes:

Enable Scrollbar (resource scrollbar)

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

Enable Jump Scroll (resource jumpscroll)

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

Enable Reverse Video (resource reversevideo)

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

Enable Auto Wraparound (resource autowrap)

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

Enable Reverse Wraparound (resource reversewrap)

Enable (or disable) reverse wraparound. This corresponds to the -rw option and the reverseWrap resource.

Enable Auto Linefeed (resource autolinefeed)

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

Enable Application Cursor Keys (resource appcursor)

Enable (or disable) application cursor keys. This corresponds to the `appcursorDefault` resource. There is no corresponding command-line option.

Enable Application Keypad (resource `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 (resource `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.

As a special case, the XON / XOFF keys (control/S and control/Q) are ignored.

Scroll to Bottom on Tty Output (resource `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 (resource `allow132`)

Enable (or disable) switching between 80 and 132 columns. This corresponds to the `-132` option and the `c132` resource.

Keep Selection (resource `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 for a given atom, it can provide the corresponding text to other clients which request the selection using that atom.

This corresponds to the `keepSelection` resource. There is no corresponding command-line option.

Telling xterm to not disown the selection does not

prevent other applications from taking ownership of the selection. When that happens, xterm receives notification that this has happened, and removes its highlighting.

See SELECT/PASTE for more information.

Select to Clipboard (resource 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.

The keepSelection resource setting applies to CLIPBOARD selections just as it does for PRIMARY selections.

However some window managers treat the clipboard specially. For instance, XQuartz's synchronization between the OSX pasteboard and the X11 clipboard causes applications to lose the selection ownership for that atom when a selection is copied to the clipboard.

See SELECT/PASTE for more information.

Enable Visual Bell (resource 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 (resource bellsUrgent)

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

Enable Pop on Bell (resource 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 (resource cursorblink)

Enable (or disable) the blinking-cursor feature. This

corresponds to the `-bc` option and the `cursorBlink` resource. There are also escape sequences (see Xterm Control Sequences):

? If the `cursorBlinkXOR` resource is set, the menu entry and the escape sequence states will be XOR'd: if both are enabled, the cursor will not blink, if only one is enabled, the cursor will blink.

? If the `cursorBlinkXOR` is not set; if either the menu entry or the escape sequence states are set, the cursor will blink.

In either case, the checkbox for the menu shows the state of the `cursorBlink` resource, which may not correspond to what the cursor is actually doing.

Enable Alternate Screen Switching (resource `titelInhibit`)

Enable (or disable) switching between the normal and alternate screens. This corresponds to the `titelInhibit` resource. There is no corresponding command-line option.

Enable Active Icon (resource `activeicon`)

Enable (or disable) the active-icon feature. This corresponds to the `-ai` option and the `activeIcon` resource.

Sixel Scrolling (resource `sixelScrolling`)

When enabled, sixel graphics are positioned at the current text cursor location, scroll the image vertically if larger than the screen, and leave the text cursor at the start of the next complete line after the image when returning to text mode (this is the default). When disabled, sixel graphics are positioned at the upper left of the screen, are cropped to fit the screen, and do not affect the text cursor location. This corresponds to the `sixelScrolling` resource. There is no corresponding command-line option.

If xterm is configured to support ReGIS graphics, this controls whether a private color palette can be used. When enabled, each graphic image uses a separate set of color registers, so that it essentially has a private palette (this is the default). If it is not set, all graphics images share a common set of registers which is how sixel and ReGIS graphics worked on actual hardware. The default is likely a more useful mode on modern TrueColor hardware. This corresponds to the `privateColorRegisters` resource. There is no corresponding command-line option.

VTxxx Commands:

Do Soft Reset (resource `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 (resource `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 (resource `clearsavedlines`)

Perform a full reset, and also clear the saved lines.

Commands for setting the current screen:

Show Tek Window (resource `tekshow`)

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

Switch to Tek Mode (resource 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 (resource 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 (resource 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 xterm fontMenu pops up when the ?control? key and pointer button three are pressed in a window. It sets the font used in the VTxxx window, or modifies the way the font is specified or displayed. There are several sections.

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

Default (resource fontdefault)

Set the font to the default, i.e., that given by the *VT100.font resource.

Unreadable (resource font1)

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

Tiny (resource font2)

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

Small (resource font3)

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

Medium (resource font4)

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

Large (resource font5)

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

Huge (resource font6)

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

Enormous (resource font7)

Set the font to that given by the *VT100.font7 resource.

Escape Sequence (resource fontescape)

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

Selection (resource 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:

Bold Fonts (resource allow-bold-fonts)

This is normally checked (enabled). When unchecked, xterm will not use bold fonts. The setting corresponds to the allowBoldFonts resource.

Line-Drawing Characters (resource 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.

Packed Font (resource font-packed)

When set, tells xterm to use the minimum glyph-width from a font when displaying characters. Use the maximum width (unchecked) to help display proportional fonts. Compare to the forcePackedFont resource.

Doublesized Characters (resource 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 (resource 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 Encoding (resource `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. It corresponds to the `utf8` resource.

UTF-8 Fonts (resource `utf8-fonts`)

This controls whether `xterm` uses UTF-8 fonts for display. It is useful for temporarily switching `xterm` to display text from an application which does not follow the locale settings. It combines the `utf8` and `utf8Fonts` resources, subject to the locale resource.

UTF-8 Titles (resource `utf8-title`)

This controls whether `xterm` accepts UTF-8 encoding for title control sequences. It corresponds to the `utf8Fonts` resource.

Initially the checkmark is set according to both the `utf8` and `utf8Fonts` resource values. If the latter is set to `?always?`, the checkmark is disabled. Likewise, if there are no fonts given in the `utf8Fonts` subresources, then the checkmark also is disabled.

The standard `XTerm` `app-defaults` file defines both sets of fonts, while the `UXTerm` `app-defaults` file defines only one set. Assuming the standard `app-defaults` files, this command will launch `xterm` able to switch between UTF-8 and ISO-8859-1 encoded fonts:

```
uxterm -class XTerm
```

The fourth section allows you to enable or disable special operations which can be controlled by writing escape sequences to the terminal.

These are disabled if the `SendEvents` feature is enabled:

Allow Color Ops (resource allow-font-ops)

This corresponds to the allowColorOps resource. Enable or disable control sequences that set/query the colors.

Allow Font Ops (resource allow-font-ops)

This corresponds to the allowFontOps resource. Enable or disable control sequences that set/query the font.

Allow Mouse Ops (resource allow-mouse-ops)

Enable or disable control sequences that cause the terminal to send escape sequences on pointer-clicks and movement. This corresponds to the allowMouseOps resource.

Allow Tcap Ops (resource allow-tcap-ops)

Enable or disable control sequences that query the terminal's notion of its function-key strings, as termcap or terminfo capabilities. This corresponds to the allowTcapOps resource.

Allow Title Ops (resource allow-title-ops)

Enable or disable control sequences that modify the window title or icon name. This corresponds to the allowTitleOps resource.

Allow Window Ops (resource allow-window-ops)

Enable or disable extended window control sequences (as used in dtterm). This corresponds to the allowWindowOps resource.

Tek Options

The xterm 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 (resource tektextlarge)

#2 Size Characters (resource tektext2)

#3 Size Characters (resource tektext3)

Small Characters (resource tektextsmall)

Commands:

PAGE (resource tekpage)

Simulates the Tektronix ?PAGE? button by

- ? clearing the window,
- ? cancelling the graphics input-mode, and
- ? moving the cursor to the home position.

RESET (resource tekreset)

Unlike the similarly-named Tektronix ?RESET? button, this does everything that PAGE does as well as resetting the line-type and font-size to their default values.

COPY (resource tekcopy)

Simulates the Tektronix ?COPY? button (which makes a hard-copy of the screen) by writing the information to a text file.

Windows:

Show VT Window (resource vtshow)

Switch to VT Mode (resource vtmode)

Hide Tek Window (resource 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 X input extension for instance allows an application to bypass all of the other (limited) authorization and security features, including the `GrabKeyboard` protocol.
- ? 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.

Subject to 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 `Enable 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 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. For example, 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 entries 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.

KEY BINDINGS

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.

Actions

The following actions are provided for use within the vt100 or tek4014 translations resources:

allow-bold-fonts(on/off/toggle)

This action sets, unsets or toggles the allowBoldFonts resource and is also invoked by the allow-bold-fonts entry in fontMenu.

allow-color-ops(on/off/toggle)

This action sets, unsets or toggles the allowColorOps resource and is also invoked by the allow-color-ops entry in fontMenu.

allow-font-ops(on/off/toggle)

This action sets, unsets or toggles the allowFontOps resource and is also invoked by the allow-font-ops entry in fontMenu.

allow-mouse-ops(on/off/toggle)

This action sets, unsets or toggles the allowMouseOps resource and is also invoked by the allow-mouse-ops entry in fontMenu.

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

This action sets, unsets or toggles the allowSendEvents resource and is also invoked by the allowsends entry in mainMenu.

allow-tcap-ops(on/off/toggle)

This action sets, unsets or toggles the allowTcapOps resource and is also invoked by the allow-tcap-ops entry in fontMenu.

allow-title-ops(on/off/toggle)

This action sets, unsets or toggles the allowTitleOps resource and is also invoked by the allow-title-ops entry in fontMenu.

allow-window-ops(on/off/toggle)

This action sets, unsets or toggles the allowWindowOps resource and is also invoked by the allow-window-ops entry in fontMenu.

alt-sends-escape()

This action toggles the state of the altSendsEscape 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()` 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.

`copy-selection(destname [, ...])`

This action puts the currently selected text into all of the selections or cutbuffers specified by `destname`. Unlike `select-end`, it does not send a mouse position or otherwise modify the internal selection state.

`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 scrollbar 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 bell. 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 deletelsDEL 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:

```
^X ESC G <line+? ?> <col+? ?>
```

dump-html()

Invokes the XHTML Screen Dump feature.

dump-svg()

Invokes the SVG Screen Dump feature.

exec-formatted(format, sourcename [, ...])

Execute an external command, using the current selection for part of the command's parameters. The first parameter, format gives the basic command. Succeeding parameters specify the selection source as in insert-selection.

The format parameter allows these substitutions:

%% inserts a "%".

%P the screen-position at the beginning of the highlighted region, as a semicolon-separated pair of integers using the values that the CUP control sequence would use.

%p the screen-position after the beginning of the highlighted region, using the same convention as ?%P?.

%S the length of the string that ?%s? would insert.

%s the content of the selection, unmodified.

%T the length of the string that ?%t? would insert.

%t the selection, trimmed of leading/trailing whitespace.

Embedded spaces (and newlines) are copied as is.

%R the length of the string that ?%r? would insert.

%r the selection, trimmed of trailing whitespace.

%V the video attributes at the beginning of the highlighted region, as a semicolon-separated list of integers using the values that the SGR control sequence would use.

%v the video attributes after the end of the highlighted region, using the same convention as ?%V?.

After constructing the command-string, xterm forks a subprocess and executes the command, which completes independently of xterm.

For example, this translation would invoke a new xterm process to view a file whose name is selected while holding the shift key down. The new process is started when the mouse button is released:

```
*VT100*translations: #override Shift \  
    <Btn1Up>:exec-formatted("xterm -e view '%t'", SELECT)
```

exec-selectable(format, onClicks)

Execute an external command, using data copied from the screen for part of the command's parameters. The first parameter, format gives the basic command as in exec-formatted. The second parameter specifies the method for copying the data as in the on2Clicks resource.

fullscreen(on/off/toggle)

This action sets, unsets or toggles the fullscreen resource.

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.

Only single-byte values are treated specially. The exact action depends on the value of the `altSendsEscape` and the `metaSendsEscape` and the `eightBitInput` resources. The `metaSendsEscape` resource is tested first. See the `eightBitInput` resource for a full discussion.

The term "eight-bit" is misleading: `xterm` checks if the key is in the range 128 to 255 (the eighth bit is set). If the value is in that range, depending on the resource values, `xterm` may then do one of the following:

? add 128 to the value, setting its eighth bit,

? send an ESC byte before the key, or

? send the key unaltered.

`insert-formatted(format, sourcename [, ...])`

Insert the current selection or data related to it, formatted.

The first parameter, `format` gives the template for the data as in `exec-formatted`. Succeeding parameters specify the selection source as in `insert-selection`.

`insert-selectable(format, onClicks)`

Insert data copied from the screen, formatted. The first parameter, `format` gives the template for the data as in `exec-formatted`. The second parameter specifies the method for copying the data as in the `on2Clicks` resource.

`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?` (i.e., `nameKeymap`, where 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, including `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.

`pointer-button()`

Use this action as a fall-back to handle button press- and release-events for the mouse control sequence protocol when the selection-related translations are suppressed with the `omitTranslation` resource.

pointer-motion()

Use this action as a fall-back to handle motion-events for the mouse control sequence protocol when the selection-related translations are suppressed with the omitTranslation resource.

popup-menu(menuName)

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

print(printer-flags)

This action prints the window. It is also invoked by the print entry in mainMenu.

The action accepts optional parameters, which temporarily override resource settings. The parameter values are matched ignoring case:

noFormFeed

no form feed will be sent at the end of the last line printed (i.e., printerFormFeed is ?false?).

FormFeed

a form feed will be sent at the end of the last line printed (i.e., printerFormFeed is ?true?).

noNewLine

no newline will be sent at the end of the last line printed, and wrapped lines will be combined into long lines (i.e., printerNewLine is ?false?).

NewLine

a newline will be sent at the end of the last line printed, and each line will be limited (by adding a newline) to the screen width (i.e., printerNewLine is ?true?).

noAttrs

the page is printed without attributes (i.e., printAttributes is ?0?).

monoAttrs

the page is printed with monochrome (vt220) attributes
(i.e., printAttributes is ?1?).

colorAttrs

the page is printed with ANSI color attributes (i.e.,
printAttributes is ?2?).

print-everything(printer-flags)

This action sends the entire text history, in addition to the text currently visible, to the program given in the printerCommand resource. It allows the same optional parameters as the print action. With a suitable printer command, the action can be used to load the text history in an editor.

print-immediate()

Sends the text of the current window directly to a file, as specified by the printFileImmediate, printModelImmediate and printOptImmediate resources.

print-on-error()

Toggles a flag telling xterm that if it exits with an X error, to send the text of the current window directly to a file, as specified by the printFileOnError, printModeOnError and printOptOnError resources.

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.

readline-button()

Supports the optional readline feature by echoing repeated cursor forward or backward control sequences on button release event, to request that the host application update its notion

of the cursor's position to match the button event.

redraw()

This action redraws the window. It 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. If no count parameter is given, xterm uses the number of lines given by the scrollLines resource.

An adjustment can be specified for the page or halfpage units by appending a `?+?` or `?-?` sign followed by a number, e.g., `page-2` to specify 2 lines less than a page.

If the second parameter is omitted `?lines?` is used.

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 (see SECURITY), and is invoked from the securekbd entry in mainMenu.

scroll-lock(on/off/toggle)

This action sets, unsets or toggles internal state which tells xterm whether Scroll Lock is active, subject to the allowScrollLock resource.

scroll-to(count)

Scroll to the given line relative to the beginning of the saved-lines. For instance, `?scroll-to(0)?` would scroll to the

beginning. Two special nonnumeric parameters are recognized:

`scroll-to(begin)`

Scroll to the beginning of the saved lines.

`scroll-to(end)`

Scroll to the end of the saved lines, i.e., to the currently active page.

`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`. It also sends a mouse position and updates the internal selection state to reflect the end of the selection process.

`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). It is also invoked by the suspend, continue, interrupt, hangup, terminate, and kill entries in mainMenu.

Allowable signal names are (case is not significant): tstp (if supported by the operating system), suspend (same as tstp), cont (if supported by the operating system), int, hup, term, quit, alarm, alarm (same as alarm) and kill.

set-8-bit-control(on/off/toggle)

This action sets, unsets or toggles the eightBitControl resource. It is also invoked from the 8-bit-control entry in vtMenu.

set-allow132(on/off/toggle)

This action sets, unsets or toggles the c132 resource. It is also invoked from the allow132 entry in vtMenu.

set-altscreen(on/off/toggle)

This action sets, unsets or toggles between the alternate and current screens.

set-appcursor(on/off/toggle)

This action sets, unsets or toggles the handling Application Cursor Key mode and is also invoked by the appcursor entry in vtMenu.

set-appkeypad(on/off/toggle)

This action sets, unsets or toggles the handling of Application Keypad mode and is also invoked by the appkeypad entry in vtMenu.

set-autolinefeed(on/off/toggle)

This action sets, unsets or toggles automatic insertion of line feeds. It is also invoked by the autolinefeed entry in vtMenu.

set-autowrap(on/off/toggle)

This action sets, unsets or toggles automatic wrapping of long lines. It is also invoked by the autowrap entry in vtMenu.

set-backarrow(on/off/toggle)

This action sets, unsets or toggles the backarrowKey resource. It is also invoked from the backarrow key entry in vtMenu.

set-bellsUrgent(on/off/toggle)

This action sets, unsets or toggles the bellsUrgent resource.

It is also invoked by the bellsUrgent entry in vtMenu.

set-cursorblink(on/off/toggle)

This action sets, unsets or toggles the cursorBlink resource.

It is also invoked from the cursorblink entry in vtMenu.

set-cursesemul(on/off/toggle)

This action sets, unsets or toggles the curses resource. It is also invoked from the cursesemul entry in vtMenu.

set-font-doublesize(on/off/toggle)

This action sets, unsets or toggles the fontDoublesize resource. It is also invoked by the font-doublesize entry in fontMenu.

set-hp-function-keys(on/off/toggle)

This action sets, unsets or toggles the hpFunctionKeys resource. It is also invoked by the hpFunctionKeys entry in mainMenu.

set-jumpscroll(on/off/toggle)

This action sets, unsets or toggles the jumpscroll resource.

It is also invoked by the jumpscroll entry in vtMenu.

set-font-linedrawing(on/off/toggle)

This action sets, unsets or 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-font-packed(on/off/toggle)

This action sets, unsets or toggles the forcePackedFont resource which controls use of the font's minimum or maximum glyph width. It is also invoked by the font-packed entry in fontMenu.

set-keep-clipboard(on/off/toggle)

This action sets, unsets or toggles the keepClipboard resource.

set-keep-selection(on/off/toggle)

This action sets, unsets or toggles the keepSelection resource.

It is also invoked by the keepSelection entry in vtMenu.

set-logging(on/off/toggle)

This action sets, unsets or toggles the state of the logging option.

set-old-function-keys(on/off/toggle)

This action sets, unsets or toggles the state of legacy function keys. It is also invoked by the oldFunctionKeys entry in mainMenu.

set-marginbell(on/off/toggle)

This action sets, unsets or toggles the marginBell resource.

set-num-lock(on/off/toggle)

This action toggles the state of the numLock resource.

set-pop-on-bell(on/off/toggle)

This action sets, unsets or toggles the popOnBell resource. It is also invoked by the poponbell entry in vtMenu.

set-private-colors(on/off/toggle)

This action sets, unsets or toggles the privateColorRegisters resource.

set-render-font(on/off/toggle)

This action sets, unsets or toggles the renderFont resource.

It is also invoked by the render-font entry in fontMenu.

set-reverse-video(on/off/toggle)

This action sets, unsets or toggles the reverseVideo resource.

It is also invoked by the reversevideo entry in vtMenu.

set-reversewrap(on/off/toggle)

This action sets, unsets or toggles the reverseWrap resource.

It is also invoked by the reversewrap entry in vtMenu.

set-scroll-on-key(on/off/toggle)

This action sets, unsets or toggles the scrollKey resource. It is also invoked from the scrollkey entry in vtMenu.

set-scroll-on-tty-output(on/off/toggle)

This action sets, unsets or toggles the scrollTtyOutput

resource. It is also invoked from the scrollttyoutput entry in vtMenu.

set-scrollbar(on/off/toggle)

This action sets, unsets or toggles the scrollbar resource. It is also invoked by the scrollbar entry in vtMenu.

set-sco-function-keys(on/off/toggle)

This action sets, unsets or toggles the scoFunctionKeys resource. It is also invoked by the scoFunctionKeys entry in mainMenu.

set-select(on/off/toggle)

This action sets, unsets or toggles the selectToClipboard resource. It is also invoked by the selectToClipboard entry in vtMenu.

set-sixel-scrolling(on/off/toggle)

This action toggles between inline (sixel scrolling) and absolute positioning. It can also be controlled via DEC private mode 80 (DECSDM) or from the sixelScrolling entry in the btMenu.

set-sun-function-keys(on/off/toggle)

This action sets, unsets or toggles the sunFunctionKeys resource. It is also invoked by the sunFunctionKeys entry in mainMenu.

set-sun-keyboard(on/off/toggle)

This action sets, unsets or toggles the sunKeyboard resource. It is also invoked by the sunKeyboard entry in mainMenu.

set-tek-text(large/2/3/small)

This action sets the font used in the Tektronix window to the value of the selected resource according to the argument. The argument can be either a keyword or single-letter alias, as shown in parentheses:

large (l)

Use resource fontLarge, same as menu entry tektextrlarge.

two (2)

Use resource font2, same as menu entry tekttext2.

three (3)

Use resource font3, same as menu entry tekttext3.

small (s)

Use resource fontSmall, same as menu entry tekttextsmall.

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-titleInhibit(on/off/toggle)

This action sets, unsets or toggles the titleInhibit resource, which controls switching between the alternate and current screens.

set-toolbar(on/off/toggle)

This action sets, unsets or toggles the toolbar feature. It is also invoked by the toolbar entry in mainMenu.

set-utf8-fonts(on/off/toggle)

This action sets, unsets or toggles the utf8Fonts resource. It is also invoked by the utf8-fonts entry in fontMenu.

set-utf8-mode(on/off/toggle)

This action sets, unsets or toggles the utf8 resource. It is also invoked by the utf8-mode entry in fontMenu.

set-utf8-title(on/off/toggle)

This action sets, unsets or toggles the utf8Title resource. It is also invoked by the utf8-title entry in fontMenu.

set-visibility(vt/tek,on/off/toggle)

This action sets, unsets or toggles 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 sets, unsets or toggles the visualBell resource. It is also invoked by the visualbell entry in vtMenu.

set-vt-font(d/1/2/3/4/5/6/7/e/s [,normalfont [, boldfont]])

This action sets the font or fonts currently being used in the VTxxx 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 7 indicate the fonts specified by the font1 through font7 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. It 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. It is also invoked by the `tekpage` entry in `tekMenu`.

tek-reset()

This action resets the Tektronix window. It 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.

Default Key Bindings

The default bindings in the VTxxx window use the SELECT token, which is set by the selectToClipboard resource. These are for the vt100 widget:

Shift <KeyPress> Prior:scroll-back(1,halpage) \n\

Shift <KeyPress> Next:scroll-forw(1,halpage) \n\

Shift <KeyPress> Select:select-cursor-start() \

select-cursor-end(SELECT, CUT_BUFFER0) \n\

Shift <KeyPress> Insert:insert-selection(SELECT, CUT_BUFFER0) \n\

Alt <Key>Return:fullscreen() \n\

<KeyRelease> Scroll_Lock:scroll-lock() \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,halpage,m) \n\
    Lock Ctrl <Btn4Down>:scroll-back(1,halpage,m) \n\
Lock @Num_Lock Ctrl <Btn4Down>:scroll-back(1,halpage,m) \n\
  @Num_Lock Ctrl <Btn4Down>:scroll-back(1,halpage,m) \n\
    <Btn4Down>:scroll-back(5,line,m) \n\
    Ctrl <Btn5Down>:scroll-forw(1,halpage,m) \n\
    Lock Ctrl <Btn5Down>:scroll-forw(1,halpage,m) \n\
Lock @Num_Lock Ctrl <Btn5Down>:scroll-forw(1,halpage,m) \n\
  @Num_Lock Ctrl <Btn5Down>:scroll-forw(1,halpage,m) \n\
    <Btn5Down>:scroll-forw(5,line,m) \n\
    <BtnUp>:select-end(SELECT, CUT_BUFFER0) \n\
    <BtnMotion>:pointer-motion() \n\
    <BtnDown>:pointer-button() \n\
    <BtnUp>:pointer-button() \n\
    <BtnDown>:ignore()

```

The default bindings in the Tektronix window are analogous but less extensive. These are for the tek4014 widget:

```

~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)

```

Custom Key Bindings

You can modify the translations resource by overriding parts of it, or merging your resources with it.

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)

```

In the example, the class name VT100 is used rather than the widget name. These are different; a class name could apply to more than one widget. A leading `??` is used because the widget hierarchy above the vt100 widget depends on whether the toolbar support is compiled into xterm.

Most of the predefined translations are related to the mouse, with a

few that use some of the special keys on the keyboard. Applications use special keys (function-keys, cursor-keys, keypad-keys) with modifiers (shift, control, alt). If xterm defines a translation for a given combination of special key and modifier, that makes it unavailable for use by applications within the terminal. For instance, one might extend the use of Page Up and Page Down keys seen here:

```
Shift <KeyPress> Prior : scroll-back(1,halpage) \n\  
Shift <KeyPress> Next  : scroll-forw(1,halpage) \n\  
to the Home and End keys:
```

```
Shift <KeyPress> Home  : scroll-to(begin) \n\  
Shift <KeyPress> End   : scroll-to(end)
```

but then shift-Home and shift-End would then be unavailable to applications.

Not everyone finds the three-button mouse bindings easy to use. In a wheel mouse, the middle button might be the wheel. As an alternative, you could add a binding using shifted keys:

```
*VT100*translations:  #override \n\  
Shift <Key>Home:  copy-selection(SELECT) \n\  
Shift <Key>Insert: copy-selection(SELECT) \n\  
Ctrl Shift <Key>C: copy-selection(SELECT) \n\  
Ctrl Shift <Key>V: insert-selection(SELECT)
```

You would still use the left- and right-mouse buttons (typically 1 and 3) for beginning and extending selections.

Besides mouse problems, there are also keyboards with inconvenient layouts. Some lack a numeric keypad, making it hard to use the shifted keypad plus and minus bindings for switching between font sizes. You can work around that by assigning the actions to more readily accessed keys:

```
*VT100*translations:  #override \n\  
Ctrl <Key> +:  larger-vt-font() \n\  
Ctrl <Key> -:  smaller-vt-font()
```

The keymap feature allows you to switch between sets of translations.

The sample below shows how the keymap() action may be used to add

special keys for entering commonly-typed words:

```
*VT100.Translations: #override <Key>F13: keymap(dbx)
```

```
*VT100.dbxKeymap.translations: \
```

```
<Key>F14:    keymap(None) \n\
```

```
<Key>F17:    string("next") \n\
```

```
            string(0x0d) \n\
```

```
<Key>F18:    string("step") \n\
```

```
            string(0x0d) \n\
```

```
<Key>F19:    string("continue") \n\
```

```
            string(0x0d) \n\
```

```
<Key>F20:    string("print ") \n\
```

```
            insert-selection(PRIMARY, CUT_BUFFER0)
```

Default Scrollbar Bindings

Key bindings are normally associated with the vt100 or tek4014 widgets which act as terminal emulators. Xterm's scrollbar (and toolbar if it is configured) are separate widgets. Because all of these use the X Toolkit, they have corresponding translations resources. Those resources are distinct, and match different patterns, e.g., the differences in widget-name and number of levels of widgets which they may contain.

The scrollbar widget is a child of the vt100 widget. It is positioned on top of the vt100 widget. Toggling the scrollbar on and off causes the vt100 widget to resize.

The default bindings for the scrollbar widget use only mouse-button events:

```
<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()
```

Events which the scrollbar widget does not recognize at all are lost.

However, at startup, xterm augments these translations with the default translations used for the vt100 widget, together with the resource ?actions? which those translations use. Because the scrollbar (or menubar) widgets do not recognize these actions (but because it has a corresponding translation), they are passed on to the vt100 widget.

This augmenting of the scrollbar's translations has a few limitations:

- ? Xterm knows what the default translations are, but there is no suitable library interface for determining what customizations a user may have added to the vt100 widget. All that xterm can do is augment the scrollbar widget to give it the same starting point for further customization by the user.
- ? Events in the gap between the widgets may be lost.
- ? Compose sequences begun in one widget cannot be completed in the other, because the input methods for each widget do not share context information.

Most customizations of the scrollbar translations do not concern key bindings. Rather, users are generally more interested in changing the bindings of the mouse buttons. For example, some people prefer using the left pointer button for dragging the scrollbar thumb. That can be set up 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

Applications can send sequences of characters to the terminal to change its behavior. Often they are referred to as ?ANSI escape sequences? or just plain ?escape sequences? but both terms are misleading:

- ? ANSI x3.64 (obsolete) which was replaced by ISO 6429 (ECMA-48) gave rules for the format of these sequences of characters.
- ? While the original VT100 was claimed to be ANSI-compatible (against

x3.64), there is no freely available version of the ANSI standard to show where the VT100 differs. Most of the documents which mention the ANSI standard have additions not found in the original (such as those based on ansi.sys). So this discussion focuses on the ISO standards.

? The standard describes only sequences sent from the host to the terminal. There is no standard for sequences sent by special keys from the terminal to the host. By convention (and referring to existing terminals), the format of those sequences usually conforms to the host-to-terminal standard.

? Some of xterm's sequences do not fit into the standard scheme. Technically those are ?unspecified?. As an example, DEC Screen Alignment Test (DECALN) is this three-character sequence:

ESC # 8

? Some sequences fit into the standard format, but are not listed in the standard. These include the sequences used for setting up scrolling margins and doing forward/reverse scrolling.

? Some of the sequences (in particular, the single-character functions such as tab and backspace) do not include the escape character.

With all of that in mind, the standard refers to these sequences of characters as ?control sequences?.

Xterm Control Sequences 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.

A few examples of usage are given in this section.

Window and Icon Titles

Some scripts use echo with options -e and -n to tell the shell to interpret the string ?\e? as the escape character and to suppress a trailing newline on output. Those are not portable, nor recommended. Instead, use printf (POSIX).

For example, to set the window title to ?Hello world!?, you could use

one of these commands in a script:

```
printf '\033]2;Hello world!\033\  
printf '\033]2;Hello world!\007'  
printf '\033]2;%s\033\ "Hello world!"  
printf '\033]2;%s\007' "Hello world!"
```

The printf command interprets the octal value `\033` for escape, and (since it was not given in the format) omits a trailing newline from the output.

Some programs (such as `screen(1)`) set both window- and icon-titles at the same time, using a slightly different control sequence:

```
printf '\033]0;Hello world!\033\  
printf '\033]0;Hello world!\007'  
printf '\033]0;%s\033\ "Hello world!"  
printf '\033]0;%s\007' "Hello world!"
```

The difference is the parameter `0` in each command. Most window managers will honor either window title or icon title. Some will make a distinction and allow you to set just the icon title. You can tell `xterm` to ask for this with a different parameter in the control sequence:

```
printf '\033]1;Hello world!\033\  
printf '\033]1;Hello world!\007'  
printf '\033]1;%s\033\ "Hello world!"  
printf '\033]1;%s\007' "Hello world!"
```

Special Keys

`Xterm`, like any VT100-compatible terminal emulator, has two modes for the special keys (cursor-keys, numeric keypad, and certain function-keys):

- ? normal mode, which makes the special keys transmit ?useful? sequences such as the control sequence for cursor-up when pressing the up-arrow, and
- ? application mode, which uses a different control sequence that cannot be mistaken for the ?useful? sequences.

The main difference between the two modes is that normal mode sequences

start with CSI (escape `[]`) and application mode sequences start with SS3 (escape `O`).

The terminal is initialized into one of these two modes (usually the normal mode), based on the terminal description (termcap or terminfo).

The terminal description also has capabilities (strings) defined for the keypad mode used in curses applications.

There is a problem in using the terminal description for applications that are not intended to be full-screen curses applications: the definitions of special keys are only correct for this keypad mode. For example, some shells (unlike `ksh(1)`, which appears to be hard-coded, not even using termcap) allow their users to customize key-bindings, assigning shell actions to special keys.

? `bash(1)` allows constant strings to be assigned to functions. This is only successful if the terminal is initialized to application mode by default, because `bash` lacks flexibility in this area. It uses a (less expressive than `bash`'s) readline scripting language for setting up key bindings, which relies upon the user to statically enumerate the possible bindings for given values of `$TERM`.

? `zsh(1)` provides an analogous feature, but it accepts runtime expressions, as well as providing a `$terminfo` array for scripts. In particular, one can use the terminal database, transforming when defining a key-binding. By transforming the output so that CSI and SS3 are equated, `zsh` can use the terminal database to obtain useful definitions for its command-line use regardless of whether the terminal uses normal or application mode initially. Here is an example:

```
[[ "$terminfo[kcuu1]" == "^[O"* ]] && \  
bindkey -M viins "${terminfo[kcuu1]/O/}" \  
vi-up-line-or-history
```

Changing Colors

A few shell programs provide the ability for users to add color and other video attributes to the shell prompt strings. Users can do this

by setting `$PS1` (the primary prompt string). Again, `bash` and `zsh` have provided features not found in `ksh`. There is a problem, however: the prompt's width on the screen will not necessarily be the same as the number of characters. Because there is no guidance in the POSIX standard, each shell addresses the problem in a different way:

? `bash` treats characters within `?\[? and ?\]? as nonprinting (using no width on the screen).`

? `zsh` treats characters within `?%{? and ?%}? as nonprinting.`

In addition to the difference in syntax, the shells provide different methods for obtaining useful escape sequences:

? As noted in `Special Keys`, `zsh` initializes the `$terminfo` array with the terminal capabilities.

It also provides a function `echoti` which works like `tput(1)` to convert a terminal capability with its parameters into a string that can be written to the terminal.

? Shells lacking a comparable feature (such as `bash`) can always use the program `tput` to do this transformation.

Hard-coded escape sequences are supported by each shell, but are not recommended because those rely upon particular configurations and cannot be easily moved between different user environments.

ENVIRONMENT

`Xterm` sets several environment variables.

System Independent

Some variables are used on every system:

DISPLAY

is the display name, pointing to the X server (see `DISPLAY NAMES` in `X(7)`).

TERM

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

On some systems, you may encounter situations where the shell which you use and `xterm` are built using libraries with different terminal databases. In that situation, `xterm` may choose a

terminal description not known to the shell.

WINDOWID

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

XTERM_FILTER

is set if a locale-filter is used. The value is the pathname of the filter.

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).

System Dependent

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

COLUMNS

the width of the xterm in characters (cf: ?stty columns?).

When this variable is set, curses applications (and most terminal programs) will assume that the terminal has this many columns.

Xterm would do this for systems which have no ability to tell the size of the terminal. Those are very rare, none newer than the mid 1990s when SVR4 became prevalent.

HOME

when xterm is configured (at build-time) to update utmp.

LINES

the height of the xterm in characters (cf: ?stty rows?).

When this variable is set, curses applications (and most terminal programs) will assume that the terminal has this many lines (rows).

Xterm would do this for systems which have no ability to tell the size of the terminal. Those are very rare, none newer than the mid 1990s when SVR4 became prevalent.

LOGNAME

when xterm is configured (at build-time) to update utmp.

Your configuration may have set LOGNAME; xterm does not modify that. If it is unset, xterm will use USER if it is set. Finally, if neither is set, xterm will use the getlogin(3) function.

SHELL

when xterm is configured (at build-time) to update utmp. It is also set if you provide a valid shell name as the optional parameter.

Xterm sets this to an absolute pathname. If you have set the variable to a relative pathname, xterm may set it to a different shell pathname.

If you have set this to an pathname which does not correspond to a valid shell, xterm may unset it, to avoid confusion.

TERMCAP

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

This feature is, like LINES and COLUMNS, used rarely. It addresses the same limitation of a few older systems by providing a way for termcap-based applications to get the initial screen size.

TERMINFO

may be defined to a nonstandard location using the configure script.

WINDOW PROPERTIES

In the output from xprop(1), there are several properties.

Properties set by X Toolkit

WM_CLASS

This shows the instance name and the X resource class, passed to X Toolkit during initialization of xterm, e.g.,

```
WM_CLASS(String) = "xterm", "UXTerm"
```

WM_CLIENT_LEADER

This shows the window-id which xterm provides with an environment variable (WINDOWID), e.g.,

```
WM_CLIENT_LEADER(WINDOW): window id # 0x800023
```

WM_COMMAND

This shows the command-line arguments for xterm which are passed to X Toolkit during initialization, e.g.,

```
WM_COMMAND(String) = { "xterm", "-class", "UXTerm", "-title", "uxterm", "-u8" }
```

WM_ICON_NAME

This holds the icon title, which different window managers handle in various ways. It is set via the iconName resource.

Applications can change this using control sequences.

WM_LOCALE_NAME

This shows the result from the setlocale(3) function for the LC_CTYPE category, e.g.,

```
WM_LOCALE_NAME(String) = "en_US.UTF-8"
```

WM_NAME

This holds the window title, normally at the top of xterm's window. It is set via the title resource. Applications can change this using control sequences.

Properties set by Xterm

X Toolkit does not manage EWMH properties. Xterm does this directly.

_NET_WM_ICON_NAME

stores the icon name.

_NET_WM_NAME

stores the title string.

_NET_WM_PID

stores the process identifier for xterm's display.

Properties used by Xterm

`_NET_SUPPORTED`

Xterm checks this property on the supporting window to decide if the window manager supports specific maximizing styles. That may include other window manager hints; xterm uses the X library calls to manage those.

`_NET_SUPPORTING_WM_CHECK`

Xterm checks this to ensure that it will only update the EWMH properties for a window manager which claims EWMH compliance.

`_NET_WM_STATE`

This tells xterm whether its window has been maximized by the window manager, and if so, what type of maximizing:

`_NET_WM_STATE_FULLSCREEN`

`_NET_WM_STATE_MAXIMIZED_HORZ`

`_NET_WM_STATE_MAXIMIZED_VERT`

FILES

The actual pathnames given may differ on your system.

`/etc/shells`

contains a list of valid shell programs, used by xterm to decide if the `?SHELL?` environment variable should be set for the process started by xterm.

On systems which have the `getusershell` function, xterm will use that function rather than directly reading the file, since the file may not be present if the system uses default settings.

`/etc/utmp`

the system log file, which records user logins.

`/var/log/wtmp`

the system log file, which records user logins and logouts.

`/usr/share/X11/app-defaults/XTerm`

the xterm default application resources.

`/usr/share/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 /usr/share/X11/app-defaults/XTerm. If you do not do this, xterm uses its compiled-in default resource settings for colors.

/usr/share/pixmaps

the directory in which xterm's pixmap icon files are installed.

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

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

85 ERROR_ICEERROR
ICE I/O error

90 ERROR_SCALLOC
Alloc: calloc() failed on base

91 ERROR_SCALLOC2
Alloc: calloc() failed on rows

102 ERROR_SAVE_PTR
ScrnPointers: 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.

When connected to an input method, it is possible for xterm to hang if

the XIM server is suspended or killed.

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)`, `uxterm(1)`, `X(7)`, `pty(4)`, `tty(4)`

Xterm Control Sequences (this is the file `ctlseqs.ms`).

<https://invisible-island.net/xterm/xterm.html>

<https://invisible-island.net/xterm/manpage/xterm.html>

<https://invisible-island.net/xterm/ctlseqs/ctlseqs.html>

<https://invisible-island.net/xterm/xterm.faq.html>

<https://invisible-island.net/xterm/xterm.log.html>

X Toolkit Intrinsic's ? C Language Interface (Xt),

Joel McCormack, Paul Asente, Ralph R. Swick (1994),

Thomas E. Dickey (2019).

Inter-Client Communication Conventions Manual (ICCCM),

David Rosenthal and Stuart W. Marks (version 2.0, 1994).

Extended Window Manager Hints (EWMH),

X Desktop Group (version 1.3, 2005).

EWMH uses `UTF8_STRING` pervasively without defining it, but does mention the ICCCM. Version 2.0 of the ICCCM does not address UTF-8. That is an extension added in XFree86.

? Markus Kuhn summarized this in UTF-8 and Unicode FAQ for Unix/Linux (2001), in the section ?Is X11 ready for Unicode??

<https://www.cl.cam.ac.uk/~mgk25/unicode.html>

? Juliusz Chroboczek proposed the `UTF8_STRING` selection atom in 1999/2000, which became part of the ICCCM in XFree86.

https://www.irif.fr/~jch/software/UTF8_STRING/

An Xorg developer removed that part of the documentation in 2004 when incorporating other work from XFree86 into Xorg. The feature is still supported in Xorg, though undocumented as of 2019.

AUTHORS

Far too many people.

These contributed to the X Consortium: 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).

Beginning with XFree86, there were far more identifiable contributors.

The THANKS file in xterm's source lists 228 in September 2020. Keep in mind these: Jason Bacon, Jens Schweikhardt, Ross Combs, Stephen P. Wall, David Wexelblat, and Thomas Dickey (invisible-island.net).

Patch #366

2021-02-10

XTERM(1)