Index

NAME
SYNOPSIS
DESCRIPTION
FREQUENTLY ASKED QUESTIONS
RXVT-UNICODE VS. RXVT
OPTIONS
RESOURCES
THE SCROLLBAR
MOUSE REPORTING
THE SELECTION: SELECTING AND PASTING TEXT
CHANGING FONTS
ISO 14755 SUPPORT
LOGIN STAMP
COLOURS AND GRAPHICS
    ALPHA CHANNEL SUPPORT
ENVIRONMENT
FILES
SEE ALSO
CURRENT PROJECT COORDINATOR
AUTHORS

NAME

rxvt-unicode (ouR XVT, unicode) - (a VT102 emulator for the X window system)

SYNOPSIS

urxvt [options] [-e command [args]]

DESCRIPTION

rxvt-unicode, version **(release docs will have VERSION here)**, is a colour vt102 terminal emulator intended as an *xterm*(1) replacement for users who do not require features such as Tektronix 4014 emulation and toolkit-style configurability. As a result, **rxvt-unicode** uses much less swap space – a significant advantage on a machine serving many X sessions.

This document is also available on the World-Wide-Web at http://pod.tst.eu/http://cvs.schmorp.de/rxvt-unicode/doc/rxvt.1.pod.

FREQUENTLY ASKED QUESTIONS

See urxvt(7) (try `man 7 urxvt`) for a list of frequently asked questions and answer to them and some common problems. That document is also accessible on the World-Wide-Web at http://pod.tst.eu/http://cvs.schmorp.de/rxvt-unicode/doc/rxvt.7.pod.

RXVT-UNICODE VS. RXVT

Unlike the original rxvt, **rxvt-unicode** stores all text in Unicode internally. That means it can store and display most scripts in the world. Being a terminal emulator, however, some things are very difficult, especially cursive scripts such as arabic, vertically written scripts like mongolian or scripts requiring extremely complex combining rules, like tibetan or devanagari. Don't expect pretty output when using these scripts. Most other scripts, latin, cyrillic, kanji, thai etc. should

work fine, though. A somewhat difficult case are right-to-left scripts, such as hebrew: **rxvt-unicode** adopts the view that bidirectional algorithms belong in the application, not the terminal emulator (too many things – such as cursor-movement while editing – break otherwise), but that might change.

If you are looking for a terminal that supports more exotic scripts, let me recommend `mlterm`, which is a very user friendly, lean and clean terminal emulator. In fact, the reason rxvt-unicode was born was solely because the author couldn't get `mlterm` to use one font for latin1 and another for japanese.

Therefore another design rationale was the use of multiple fonts to display characters: The idea of a single unicode font which many other programs force onto its users never made sense to me: You should be able to choose any font for any script freely.

Apart from that, rxvt-unicode is also much better internationalised than its predecessor, supports things such as XFT and ISO 14755 that are handy in i18n-environments, is faster, and has a lot bugs less than the original rxvt. This all in addition to dozens of other small improvements.

It is still faithfully following the original rxvt idea of being lean and nice on resources: for example, you can still configure rxvt-unicode without most of its features to get a lean binary. It also comes with a client/daemon pair that lets you open any number of terminal windows from within a single process, which makes startup time very fast and drastically reduces memory usage. See urxvtd(1) (daemon) and urxvtc(1) (client).

It also makes technical information about escape sequences (which have been extended) more accessible: see urxvt(7) for technical reference documentation (escape sequences etc.).

OPTIONS

The **urxvt** options (mostly a subset of *xterm*'s) are listed below. In keeping with the smaller-is-better philosophy, options may be eliminated or default values chosen at compile-time, so options and defaults listed may not accurately reflect the version installed on your system. `urxvt -h' gives a list of major compile-time options on the *Options* line. Option descriptions may be prefixed with which compile option each is dependent upon. e.g. `Compile *XIM*:' requires *XIM* on the *Options* line. Note: `urxvt -help' gives a list of all command-line options compiled into your version.

Note that **urxvt** permits the resource name to be used as a long-option (-/++ option) so the potential command-line options are far greater than those listed. For example: `urxvt –loginShell –color1 Orange'.

The following options are available:

-help, –help

> Print out a message describing available options.

-display *displayname*

> Attempt to open a window on the named X display (the older form **-d** is still respected. but deprecated). In the absence of this option, the display specified by the **DISPLAY** environment variable is used.

-depth *bitdepth*

> Compile *frills*: Attempt to find a visual with the given bit depth; resource **depth**.
>
> [Please note that many X servers (and libXft) are buggy with respect to `-depth 32` and/or alpha channels, and will cause all sorts of graphical corruption. This is harmless, but we can't do anything about this, so watch out]

-visual *visualID*

> Compile *frills*: Use the given visual (see e.g. `xdpyinfo` for possible visual ids) instead of the default, and also allocate a private colormap. All visual types except for DirectColor are supported.

-geometry *geom*

Window geometry (**-g** still respected); resource **geometry**.

-rv|+rv

Turn on/off simulated reverse video; resource **reverseVideo**.

-j|+j

Turn on/off jump scrolling (allow multiple lines per refresh); resource **jumpScroll**.

-ss|+ss

Turn on/off skip scrolling (allow multiple screens per refresh); resource **skipScroll**.

-fade *number*

Fade the text by the given percentage when focus is lost. Small values fade a little only, 100 completely replaces all colours by the fade colour; resource **fading**.

-fadecolor *colour*

Fade to this colour when fading is used (see **-fade**). The default colour is opaque black. resource **fadeColor**.

-icon *file*

Compile *pixbuf*: Use the specified image as application icon. This is used by many window managers, taskbars and pagers to represent the application window; resource *iconFile*.

-bg *colour*

Window background colour; resource **background**.

-fg *colour*

Window foreground colour; resource **foreground**.

-cr *colour*

The cursor colour; resource **cursorColor**.

-pr *colour*

The mouse pointer foreground colour; resource **pointerColor**.

-pr2 *colour*

The mouse pointer background colour; resource **pointerColor2**.

-bd *colour*

The colour of the border around the text area and between the scrollbar and the text; resource **borderColor**.

-fn *fontlist*

Select the fonts to be used. This is a comma separated list of font names that are checked in order when trying to find glyphs for characters. The first font defines the cell size for characters; other fonts might be smaller, but not (in general) larger. A (hopefully) reasonable default font list is always appended to it. See resource **font** for more details.

In short, to specify an X11 core font, just specify its name or prefix it with `x:`. To specify an XFT-font, you need to prefix it with `xft:`, e.g.:

```
urxvt -fn "xft:Bitstream Vera Sans Mono:pixelsize=15"
urxvt -fn "9x15bold,xft:Bitstream Vera Sans Mono"
```

See also the question "How does rxvt-unicode choose fonts?" in the FAQ section of urxvt(7).

-fb *fontlist*

Compile *font-styles*: The bold font list to use when **bold** characters are to be printed. See resource **boldFont** for details.

-fi *fontlist*

Compile *font-styles*: The italic font list to use when *italic* characters are to be printed. See resource **italicFont** for details.

-fbi *fontlist*

Compile *font-styles*: The bold italic font list to use when ***bold italic*** characters are to be printed. See resource **boldItalicFont** for details.

-is|+is

Compile *font-styles*: Bold/Blink font styles imply high intensity foreground/background (default). See resource **intensityStyles** for details.

-name *name*

Specify the application name under which resources are to be obtained, rather than the default executable file name. Name should not contain `.' or `*' characters. Also sets the icon and title name.

-ls|+ls

Start as a login-shell/sub-shell; resource **loginShell**.

-mc *milliseconds*

Specify the maximum time between multi-click selections.

-ut|+ut

Compile *utmp*: Inhibit/enable writing a utmp entry; resource **utmpInhibit**.

-vb|+vb

Turn on/off visual bell on receipt of a bell character; resource **visualBell**.

-sb|+sb

Turn on/off scrollbar; resource **scrollBar**.

-sr|+sr

Put scrollbar on right/left; resource **scrollBar_right**.

-st|+st

Display rxvt (non XTerm/NeXT) scrollbar without/with a trough; resource **scrollBar_floating**.

-si|+si

Turn on/off scroll-to-bottom on TTY output inhibit; resource **scrollTtyOutput** has opposite effect.

-sk|+sk

Turn on/off scroll-to-bottom on keypress; resource **scrollTtyKeypress**.

-sw|+sw

Turn on/off scrolling with the scrollback buffer as new lines appear. This only takes effect if **-si** is also given; resource **scrollWithBuffer**.

-ptab|+ptab

If enabled (default), "Horizontal Tab" characters are being stored as actual wide characters in the screen buffer, which makes it possible to select and paste them. Since a horizontal tab is a cursor movement and not an actual glyph, this can sometimes be visually annoying as the cursor on a tab character is displayed as a wide cursor; resource **pastableTabs**.

-bc|+bc

Blink the cursor; resource **cursorBlink**.

-uc|+uc

Make the cursor underlined; resource **cursorUnderline**.

-iconic

Start iconified, if the window manager supports that option. Alternative form is **-ic**.

-sl *number*

Save *number* lines in the scrollback buffer. See resource entry for limits; resource **saveLines**.

-b *number*

Compile *frills*: Internal border of *number* pixels. See resource entry for limits; resource **internalBorder**.

-w *number*

Compile *frills*: External border of *number* pixels. Also, **-bw** and **-borderwidth**. See resource entry for limits; resource **externalBorder**.

-bl

Compile *frills*: Set MWM hints to request a borderless window, i.e. if honoured by the WM, the rxvt-unicode window will not have window decorations; resource **borderLess**. If the window manager does not support MWM hints (e.g. kwin), enables override-redirect mode.

-override-redirect

Compile *frills*: Sets override-redirect on the window; resource **override-redirect**.

-dockapp

Sets the initial state of the window to WithdrawnState, which makes window managers that support this extension treat it as a dockapp.

-sbg

Compile *frills*: Disable the usage of the built-in block graphics/line drawing characters and just rely on what the specified fonts provide. Use this if you have a good font and want to use its block graphic glyphs; resource **skipBuiltinGlyphs**.

-lsp *number*

Compile *frills*: Lines (pixel height) to insert between each row of the display. Useful to work around font rendering problems; resource **lineSpace**.

-letsp *number*

Compile *frills*: Amount to adjust the computed character width by to control overall letter spacing. Negative values will tighten up the letter spacing, positive values will space letters out more. Useful to work around odd font metrics; resource **letterSpace**.

-tn *termname*

This option specifies the name of the terminal type to be set in the **TERM** environment variable. This terminal type must exist in the *termcap(5)* database and should have *li#* and *co#* entries; resource **termName**.

-e *command [arguments]*

Run the command with its command-line arguments in the **urxvt** window; also sets the window title and icon name to be the basename of the program being executed if neither *-title* (*-T*) nor *-n* are given on the command line. If this option is used, it must be the last on the command-line. If there is no **-e** option then the default is to run the program specified by the **SHELL** environment variable or, failing that, *sh(1)*.

Please note that you must specify a program with arguments. If you want to run shell commands, you have to specify the shell, like this:

```
urxvt -e sh -c "shell commands"
```

-title *text*

Window title (**-T** still respected); the default title is the basename of the program specified after the **-e** option, if any, otherwise the application name; resource **title**.

-n *text*

Icon name; the default name is the basename of the program specified after the **-e** option, if any, otherwise the application name; resource **iconName**.

-C

Capture system console messages.

-pt *style*

Compile *XIM*: input style for input method; **OverTheSpot**, **OffTheSpot**, **Root**; resource **preeditType**.

If the perl extension `xim-onthespot` is used (which is the default), then additionally the `OnTheSpot` preedit type is available.

-im *text*

Compile *XIM*: input method name. resource **inputMethod**.

-imlocale *string*

The locale to use for opening the IM. You can use an `LC_CTYPE` of e.g. `de_DE.UTF-8` for normal text processing but `ja_JP.EUC-JP` for the input extension to be able to input japanese characters while staying in another locale. resource **imLocale**.

-imfont *fontset*

Set the font set to use for the X Input Method, see resource **imFont** for more info.

-tcw

Change the meaning of triple-click selection with the left mouse button. Only effective when the original (non-perl) selection code is in-use. Instead of selecting a full line it will extend the selection to the end of the logical line only. resource **tripleclickwords**.

-dpb|+dpb

Compile frills: Disable (or enable) emitting bracketed paste mode sequences (default enabled). Bracketed paste mode allows programs to detect when something is pasted. Since more and more programs abuse this, these sequences can be disabled. The command sequences to enable and query paste mode will still work, but the actual bracket sequences will no longer be emitted. You can also toggle this from the ctrl-middle-mouse-button menu; resource **disablePasteBrackets**.

-insecure

Enable "insecure" mode, which currently enables most of the escape sequences that echo strings. See the resource **insecure** for more info.

-mod *modifier*

Override detection of Meta modifier with specified key: **alt**, **meta**, **hyper**, **super**, **mod1**, **mod2**, **mod3**, **mod4**, **mod5**; resource *modifier*.

-ssc|+ssc

Turn on/off secondary screen (default enabled); resource **secondaryScreen**.

-ssr|+ssr

Turn on/off secondary screen scroll (default enabled); resource **secondaryScroll**.

-rm *mode*

Compile *frills*: Sets long line rewrapping behaviour on window resizes to one of **auto** (the default), **always** or **never**. The latter two modes do the obvious, **auto** rewraps (acts like **always**) if scrollback is non-empty, and wings lines (acts like **never**) otherwise; resource **rewrapMode**.

-hold|+hold

Turn on/off hold window after exit support. If enabled, urxvt will not immediately destroy its window when the program executed within it exits. Instead, it will wait till it is being killed or closed by the user; resource **hold**.

-cd *path*

Sets the working directory for the shell (or the command specified via **-e**). The *path* must be an absolute path and it must exist for urxvt to start; resource **chdir**.

-xrm *string*

Works like the X Toolkit option of the same name, by adding the *string* as if it were specified in a resource file. Resource values specified this way take precedence over all other resource specifications.

Note that you need to use the *same* syntax as in the .Xdefaults file, e.g. `*.background: black`. Also note that all urxvt-specific options can be specified as long-options on the commandline, so use of **-xrm** is mostly limited to cases where you want to specify other resources (e.g. for input methods) or for compatibility with other programs.

-keysym.*sym string*

Remap a key symbol. See resource **keysym**.

-embed *windowid*

Tells urxvt to embed its windows into an already-existing window, which enables applications to easily embed a terminal.

Right now, urxvt will first unmap/map the specified window, so it shouldn't be a top-level window. urxvt will also reconfigure it quite a bit, so don't expect it to keep some specific state. It's best to create an extra subwindow for urxvt and leave it alone.

The window will not be destroyed when urxvt exits.

It might be useful to know that urxvt will not close file descriptors passed to it (except for stdin/out/err, of course), so you can use file descriptors to communicate with the programs within the terminal. This works regardless of whether the `-embed` option was used or not.

Here is a short Gtk2-perl snippet that illustrates how this option can be used (a longer example is in *doc/embed*):

```
my $rxvt = new Gtk2::Socket;
$rxvt->signal_connect_after (realize => sub {
   my $xid = $_[0]->window->get_xid;
   system "urxvt -embed $xid &";
});
```

-pty-fd *file descriptor*

Tells urxvt NOT to execute any commands or create a new pty/tty pair but instead use the given file descriptor as the tty master. This is useful if you want to drive urxvt as a generic terminal emulator without having to run a program within it.

If this switch is given, urxvt will not create any utmp/wtmp entries and will not tinker with pty/tty permissions - you have to do that yourself if you want that.

As an extremely special case, specifying `-1` will completely suppress pty/tty operations, which is probably only useful in conjunction with some perl extension that manages the terminal.

Here is a example in perl that illustrates how this option can be used (a longer example is in *doc/pty-fd*):

```
use IO::Pty;
use Fcntl;

my $pty = new IO::Pty;
fcntl $pty, F_SETFD, 0; # clear close-on-exec
system "urxvt -pty-fd " . (fileno $pty) . "&";
close $pty;

# now communicate with rxvt
my $slave = $pty->slave;
while (<$slave>) { print $slave "got <$_>\n" }
```

Note that, despite what the name might imply, the file descriptor does not need to be a pty, it can be a bi-directional pipe as well (e.g. a unix domain or tcp socket). While tty operations cannot be done in this case, **urxvt** can still be remote controlled with it:

```
use Socket;
use Fcntl;

socketpair my $URXVT, my $slave, Socket::AF_UNIX, Socket::SOCK_STREAM, Socket::PF_UNSPEC;
fcntl $slave, Fcntl::F_SETFD, 0;
system "exec urxvt -pty-fd " . (fileno $slave) . " &";
close $slave;

syswrite $URXVT, "Type a secret password: ";
my $secret = do { local $/ = "\r"; <$URXVT> };
print "Not so secret anymore: $secret\n";
```

-pe *string*

Comma-separated list of perl extension scripts to use (or not to use) in this terminal instance. See resource **perl-ext** for details.

RESOURCES

Note: `urxvt –help' gives a list of all resources (long options) compiled into your version. All resources are also available as long-options.

You can set and change the resources using X11 tools like **xrdb**. Many distribution do also load settings from the **~/.Xresources** file when X starts. urxvt will consult the following files/resources in order, with later settings overwriting earlier ones:

```
1. app-defaults file in $XAPPLRESDIR
2. $HOME/.Xdefaults
3. RESOURCE_MANAGER property on root-window of screen 0
4. SCREEN_RESOURCES property on root-window of the current screen
5. $XENVIRONMENT file OR $HOME/.Xdefaults-<nodename>
6. resources specified via -xrm on the commandline
```

Note that when reading X resources, **urxvt** recognizes two class names: **Rxvt** and **URxvt**. The class name **Rxvt** allows resources common to both **urxvt** and the original *rxvt* to be easily configured, while the class name **URxvt** allows resources unique to **urxvt**, to be shared between different **urxvt** configurations. If no resources are specified, suitable defaults will be used. Command-line arguments can be used to override resource settings. The following resources are supported (you might want to check the urxvtperl(3) manpage for additional settings by perl extensions not documented here):

depth: *bitdepth*

Compile *xft*: Attempt to find a visual with the given bit depth; option **-depth**.

buffered: *boolean*

Compile *xft*: Turn on/off double-buffering for xft (default enabled). On some card/driver combination enabling it slightly decreases performance, on most it greatly helps it. The slowdown is small, so it should normally be enabled.

geometry: *geom*

Create the window with the specified X window geometry [default 80x24]; option **-geometry**.

background: *colour*

Use the specified colour as the window's background colour [default White]; option **-bg**.

foreground: ***colour***

Use the specified colour as the window's foreground colour [default Black]; option **-fg**.

color*n*: ***colour***

Use the specified colour for the colour value *n*, where 0-7 corresponds to low-intensity (normal) colours and 8-15 corresponds to high-intensity (bold = bright foreground, blink = bright background) colours. The canonical names are as follows: 0=black, 1=red, 2=green, 3=yellow, 4=blue, 5=magenta, 6=cyan, 7=white, but the actual colour names used are listed in the **COLOURS AND GRAPHICS** section.

Colours higher than 15 cannot be set using resources (yet), but can be changed using an escape command (see urxvt(7)).

Colours 16-79 form a standard 4x4x4 colour cube (the same as xterm with 88 colour support). Colours 80-87 are evenly spaces grey steps.

colorBD: ***colour***

colorIT: ***colour***

Use the specified colour to display bold or italic characters when the foreground colour is the default. If font styles are not available (Compile *styles*) and this option is unset, reverse video is used instead.

colorUL: ***colour***

Use the specified colour to display underlined characters when the foreground colour is the default.

underlineColor: ***colour***

If set, use the specified colour as the colour for the underline itself. If unset, use the foreground colour.

highlightColor: ***colour***

If set, use the specified colour as the background for highlighted characters. If unset, use reverse video.

highlightTextColor: ***colour***

If set and highlightColor is set, use the specified colour as the foreground for highlighted characters.

cursorColor: ***colour***

Use the specified colour for the cursor. The default is to use the foreground colour; option **-cr**.

cursorColor2: ***colour***

Use the specified colour for the colour of the cursor text. For this to take effect, **cursorColor** must also be specified. The default is to use the background colour.

reverseVideo: ***boolean***

True: simulate reverse video by foreground and background colours; option **-rv**. **False**: regular screen colours [default]; option **+rv**. See note in **COLOURS AND GRAPHICS** section.

jumpScroll: ***boolean***

True: specify that jump scrolling should be used. When receiving lots of lines, urxvt will only scroll once a whole screen height of lines has been read, resulting in fewer updates while still displaying every received line; option **-j**.

False: specify that smooth scrolling should be used. urxvt will force a screen refresh on each new line it received; option **+j**.

skipScroll: ***boolean***

True: (the default) specify that skip scrolling should be used. When receiving lots of lines, urxvt will only scroll once in a while (around 60 times per second), resulting in far fewer updates. This can result in urxvt not ever displaying some of the lines it receives; option **-ss**.

False: specify that everything is to be displayed, even if the refresh is too fast for the human eye to read anything (or the monitor to display anything); option **+ss**.

fading: ***number***

Fade the text by the given percentage when focus is lost; option **-fade**.

fadeColor: ***colour***

Fade to this colour, when fading is used (see **fading:**). The default colour is black; option **-fadecolor**.

iconFile: ***file***

Set the application icon pixmap; option **-icon**.

scrollColor: ***colour***

Use the specified colour for the scrollbar [default #B2B2B2].

troughColor: ***colour***

Use the specified colour for the scrollbar's trough area [default #969696]. Only relevant for rxvt (non XTerm/NeXT) scrollbar.

borderColor: ***colour***

The colour of the border around the text area and between the scrollbar and the text.

font: ***fontlist***

Select the fonts to be used. This is a comma separated list of font names that are checked in order when trying to find glyphs for characters. The first font defines the cell size for characters; other fonts might be smaller, but not (in general) larger. A (hopefully) reasonable default font list is always appended to it; option **-fn**.

Each font can either be a standard X11 core font (XLFD) name, with optional prefix `x:` or a Xft font (Compile *xft*), prefixed with `xft:`.

In addition, each font can be prefixed with additional hints and specifications enclosed in square brackets (`[]`). The only available hint currently is `codeset=codeset-name`, and this is only used for Xft fonts.

For example, this font resource

```
URxvt.font: 9x15bold,\
            -misc-fixed-bold-r-normal--15-140-75-75-c-90-iso10646-1,\
            -misc-fixed-medium-r-normal--15-140-75-75-c-90-iso10646-1, \
            [codeset=JISX0208]xft:Kochi Gothic:antialias=false, \
            xft:Code2000:antialias=false
```

specifies five fonts to be used. The first one is `9x15bold` (actually the iso8859-1 version of the second font), which is the base font (because it is named first) and thus defines the character cell grid to be 9 pixels wide and 15 pixels high.

The second font is just used to add additional unicode characters not in the base font, likewise the third, which is unfortunately non-bold, but the bold version of the font does contain fewer characters, so this is a useful supplement.

The third font is an Xft font with aliasing turned off, and the characters are limited to the **JIS 0208** codeset (i.e. japanese kanji). The font contains other characters, but we are not interested in them.

The last font is a useful catch-all font that supplies most of the remaining unicode characters.

boldFont: ***fontlist***

italicFont: ***fontlist***

boldItalicFont: ***fontlist***

The font list to use for displaying **bold**, *italic* or ***bold italic*** characters, respectively.

If specified and non-empty, then the syntax is the same as for the **font**-resource, and the given font list will be used as is, which makes it possible to substitute completely different font styles for bold and italic.

If unset (the default), a suitable font list will be synthesized by "morphing" the normal text font list into the desired shape. If that is not possible, replacement fonts of the desired shape will be tried.

If set, but empty, then this specific style is disabled and the normal text font will being used for the given style.

intensityStyles: ***boolean***

When font styles are not enabled, or this option is enabled (**True**, option **-is**, the default), bold/blink font styles imply high intensity foreground/background colours. Disabling this option (**False**, option **+is**) disables this behaviour, the high intensity colours are not reachable.

title: ***string***

Set window title string, the default title is the command-line specified after the **-e** option, if any, otherwise the application name; option **-title**.

iconName: ***string***

Set the name used to label the window's icon or displayed in an icon manager window, it also sets the window's title unless it is explicitly set; option **-n**.

mapAlert: ***boolean***

True: de-iconify (map) on receipt of a bell character. **False**: no de-iconify (map) on receipt of a bell character [default].

urgentOnBell: ***boolean***

True: set the urgency hint for the wm on receipt of a bell character. **False**: do not set the urgency hint [default].

urxvt resets the urgency hint on every focus change.

visualBell: ***boolean***

True: use visual bell on receipt of a bell character; option **-vb**. **False**: no visual bell [default]; option **+vb**.

loginShell: ***boolean***

True: start as a login shell by prepending a `-' to **argv[0]** of the shell; option **-ls**. **False**: start as a normal sub-shell [default]; option **+ls**.

multiClickTime: ***number***

Specify the maximum time in milliseconds between multi-click select events. The default is 500 milliseconds; option **-mc**.

utmpInhibit: ***boolean***

True: inhibit writing record into the system log file **utmp**; option **-ut**. **False**: write record into the system log file **utmp** [default]; option **+ut**.

print-pipe: ***string***

Specify a command pipe for vt100 printer [default *lpr(1)*]. Use **Print** to initiate a screen dump to the printer and **Ctrl-Print** or **Shift-Print** to include the scrollback as well.

The string will be interpreted as if typed into the shell as-is.

Example:

```
URxvt.print-pipe: cat > $(TMPDIR=$HOME mktemp urxvt.XXXXXX)
```

This creates a new file in your home directory with the screen contents every time you hit `Print`.

scrollstyle: ***mode***

Set scrollbar style to **rxvt**, **plain**, **next** or **xterm**. **plain** is the author's favourite.

thickness: ***number***

Set the scrollbar width in pixels.

scrollBar: ***boolean***

True: enable the scrollbar [default]; option **-sb**. **False**: disable the scrollbar; option **+sb**.

scrollBar_right: ***boolean***

True: place the scrollbar on the right of the window; option **-sr**. **False**: place the scrollbar on the left of the window; option **+sr**.

scrollBar_floating: ***boolean***

True: display an rxvt scrollbar without a trough; option **-st**. **False**: display an rxvt scrollbar with a trough; option **+st**.

scrollBar_align: ***mode***

Align the **top**, **bottom** or **centre** [default] of the scrollbar thumb with the pointer on middle button press/drag.

scrollTtyOutput: ***boolean***

True: scroll to bottom when tty receives output; option **-si**. **False**: do not scroll to bottom when tty receives output; option **+si**.

scrollWithBuffer: ***boolean***

True: scroll with scrollback buffer when tty receives new lines (i.e. try to show the same lines) and **scrollTtyOutput** is False; option **-sw**. **False**: do not scroll with scrollback buffer when tty receives new lines; option **+sw**.

scrollTtyKeypress: ***boolean***

True: scroll to bottom when a non-special key is pressed. Special keys are those which are intercepted by rxvt-unicode for special handling and are not passed onto the shell; option **-sk**. **False**: do not scroll to bottom when a non-special key is pressed; option **+sk**.

saveLines: ***number***

Save *number* lines in the scrollback buffer [default 1000]; option **-sl**.

internalBorder: ***number***

Internal border of *number* pixels. This resource is limited to 100; option **-b**.

externalBorder: ***number***

External border of *number* pixels. This resource is limited to 100; option **-w**, **-bw**, **-borderwidth**.

borderLess: ***boolean***

Set MWM hints to request a borderless window, i.e. if honoured by the WM, the rxvt-unicode window will not have window decorations; option **-bl**.

skipBuiltinGlyphs: ***boolean***

Compile *frills*: Disable the usage of the built-in block graphics/line drawing characters and just rely on what the specified fonts provide. Use this if you have a good font and want to use its block graphic glyphs; option **-sbg**.

termName: ***termname***

Specifies the terminal type name to be set in the **TERM** environment variable; option **-tn**.

lineSpace: ***number***

Specifies number of lines (pixel height) to insert between each row of the display [default 0]; option **-lsp**.

meta8: ***boolean***

True: handle Meta (Alt) + keypress to set the 8th bit. **False**: handle Meta (Alt) + keypress as an escape prefix [default].

mouseWheelScrollPage: ***boolean***

True: the mouse wheel scrolls a page full. **False**: the mouse wheel scrolls five lines [default].

pastableTabs: ***boolean***

True: store tabs as wide characters. **False**: interpret tabs as cursor movement only; option `-ptab`.

cursorBlink: ***boolean***

True: blink the cursor. **False**: do not blink the cursor [default]; option **-bc**.

cursorUnderline: ***boolean***

True: Make the cursor underlined. **False**: Make the cursor a box [default]; option **-uc**.

pointerBlank: ***boolean***

True: blank the pointer when a key is pressed or after a set number of seconds of inactivity. **False**: the pointer is always visible [default].

pointerColor: ***colour***

Mouse pointer foreground colour.

pointerColor2: ***colour***

Mouse pointer background colour.

pointerShape: ***string***

Compile *frills*: Specifies the name of the mouse pointer shape [default **xterm**]. See the macros in the **X11/cursorfont.h** include file for possible values (omit the `XC_` prefix).

pointerBlankDelay: ***number***

Specifies number of seconds before blanking the pointer [default 2]. Use a large number (e.g. `987654321`) to effectively disable the timeout.

backspacekey: ***string***

The string to send when the backspace key is pressed. If set to **DEC** or unset it will send **Delete** (code 127) or, with control, **Backspace** (code 8) - which can be reversed with the appropriate DEC private mode escape sequence.

deletekey: ***string***

The string to send when the delete key (not the keypad delete key) is pressed. If unset it will send the sequence traditionally associated with the **Execute** key.

cutchars: ***string***

The characters used as delimiters for double-click word selection (whitespace delimiting is added automatically if resource is given).

When the perl selection extension is in use (the default if compiled in, see the urxvtperl(3) manpage), a suitable regex using these characters will be created (if the resource exists, otherwise, no regex will be created). In this mode, characters outside ISO-8859-1 can be used.

When the selection extension is not used, only ISO-8859-1 characters can be used. If not specified, the built-in default is used:

BACKSLASH `"'&()*,;<=>?@[]^{|}

preeditType: ***style***

OnTheSpot, **OverTheSpot**, **OffTheSpot**, **Root**; option **-pt**.

inputMethod: ***name***

name of inputMethod to use; option **-im**.

imLocale: ***name***

The locale to use for opening the IM. You can use an `LC_CTYPE` of e.g. `de_DE.UTF-8` for normal text processing but `ja_JP.EUC-JP` for the input extension to be able to input japanese characters while staying in another locale; option **-imlocale**.

imFont: ***fontset***

Specify the font-set used for XIM styles `OverTheSpot` or `OffTheSpot`. It must be a standard X font set (XLFD patterns separated by commas), i.e. it's not in the same format as the other font lists used in urxvt. The default will be set-up to chose *any* suitable found found, preferably one or two pixels differing in size to the base font. option **-imfont**.

tripleclickwords: ***boolean***

Change the meaning of triple-click selection with the left mouse button. Instead of selecting a full line it will extend the selection to the end of the logical line only; option **-tcw**.

disablePasteBrackets: ***boolean***

Prevent emission of paste bracket sequences; option **-dpb**.

insecure: ***boolean***

Enable "insecure" mode. Rxvt-unicode offers some escape sequences that echo arbitrary strings like the icon name or the locale. This could be abused if somebody gets 8-bit-clean access to your display, whether through a mail client displaying mail bodies unfiltered or through write(1) or any other means. Therefore, these sequences are disabled by default. (Note that many other terminals, including xterm, have these sequences enabled by default, which doesn't make it safer, though).

You can enable them by setting this boolean resource or specifying **-insecure** as an option. At the moment, this enables display-answer, locale, findfont, icon label and window title requests.

modifier: ***modifier***

Set the key to be interpreted as the Meta key to: **alt**, **meta**, **hyper**, **super**, **mod1**, **mod2**, **mod3**, **mod4**, **mod5**; option **-mod**.

answerbackString: ***string***

Specify the reply rxvt-unicode sends to the shell when an ENQ (control-E) character is passed through. It may contain escape values as described in the entry on **keysym** following.

secondaryScreen: ***boolean***

Turn on/off secondary screen (default enabled).

rewrapMode: ***mode***

Sets long line rewrap behaviour on window resize to one of **auto** (default), **always** or **never**.

secondaryScroll: ***boolean***

Turn on/off secondary screen scroll (default enabled). If this option is enabled, scrolls on the secondary screen will change the scrollback buffer and, when secondaryScreen is off, switching to/from the secondary screen will instead scroll the screen up.

hold: ***boolean***

Turn on/off hold window after exit support. If enabled, urxvt will not immediately destroy its window when the program executed within it exits. Instead, it will wait till it is being killed or closed by the user.

chdir: ***path***

Sets the working directory for the shell (or the command specified via **-e**). The *path* must be an absolute path and it must exist for urxvt to start. If it isn't specified then the current working directory will be used; option **-cd**.

keysym.*sym*: ***action***

Compile *frills*: Associate *action* with keysym *sym*. The intervening resource name **keysym.** cannot be omitted.

Using this resource, you can map key combinations such as `Ctrl-Shift-BackSpace` to various actions, such as outputting a different string than would normally result from that combination, making the terminal scroll up or down the way you want it, or any other thing an extension might provide.

The key combination that triggers the action, *sym*, has the following format:

```
(modifiers-)key
```

Where *modifiers* can be any combination of the following full or abbreviated modifier names:

ISOLevel3	I
AppKeypad	K
Control	C
NumLock	N
Shift	S
Meta	M or A
Lock	L
Mod1	1
Mod2	2
Mod3	3
Mod4	4
Mod5	5

The **NumLock**, **Meta** and **ISOLevel3** modifiers are usually aliased to whatever modifier the NumLock key, Meta/Alt keys or ISO Level3 Shift/AltGr keys are being mapped. **AppKeypad** is a synthetic modifier mapped to the current application keymap mode state.

Due the the large number of modifier combinations, a key mapping will match if *at least* the specified identifiers are being set, and no other key mappings with those and more bits are being defined. That means that defining a mapping for `a` will automatically provide definitions for `Meta-a`, `Shift-a` and so on, unless some of those are defined mappings themselves. See the `builtin:` action, below, for a way to work around this when this is a problem.

The spelling of *key* depends on your implementation of X. An easy way to find a key name is to use the **xev**(1) command. You can find a list by looking for the `XK_` macros in the **X11/keysymdef.h** include file (omit the `XK_` prefix). Alternatively you can specify *key* by its hex keysym value (**0x0000 - 0xFFFF**).

As with any resource value, the *action* string may contain backslash escape sequences (`\n`: newline, `\\`: backslash, `\000`: octal number), see RESOURCES in `man 7 X` for further details.

An action starts with an action prefix that selects a certain type of action, followed by a colon. An action string without colons is interpreted as a literal string to pass to the tty (as if it was prefixed with `string:`).

The following action prefixes are known - extensions can provide additional prefixes:

string:STRING

If the *action* starts with `string:` (or otherwise contains no colons), then the remaining `STRING` will be passed to the program running in the terminal. For example, you could replace whatever Shift-Tab outputs by the string `echo rm -rf /` followed by a newline:

```
URxvt.keysym.Shift-Tab: string:echo rm -rf /\n
```

This could in theory be used to completely redefine your keymap.

In addition, for actions of this type, you can define a range of keysyms in one shot by loading the `keysym-list` perl extension and providing an *action* with pattern **list/PREFIX/MIDDLE/SUFFIX**, where the delimiter `/' should be a character not used by the strings.

Its usage can be demonstrated by an example:

```
URxvt.keysym.M-C-0x61:    list|\033<|abc|>
```

The above line is equivalent to the following three lines:

```
URxvt.keysym.Meta-Control-0x61:    string:\033<a>
URxvt.keysym.Meta-Control-0x62:    string:\033<b>
```

```
URxvt.keysym.Meta-Control-0x63:    string:\033<c>
```

command:STRING

If *action* takes the form of `command:STRING`, the specified **STRING** is interpreted and executed as urxvt's control sequence (basically the opposite of `string:` - instead of sending it to the program running in the terminal, it will be treated as if it were program output). This is most useful to feed command sequences into urxvt.

For example the following means "change the current locale to `zh_CN.GBK` when Control-Meta-c is being pressed":

```
URxvt.keysym.M-C-c: command:\033]701;zh_CN.GBK\007
```

The following example will map Control-Meta-1 and Control-Meta-2 to the fonts `suxuseuro` and `9x15bold`, so you can have some limited font-switching at runtime:

```
URxvt.keysym.M-C-1: command:\033]50;suxuseuro\007
URxvt.keysym.M-C-2: command:\033]50;9x15bold\007
```

Other things are possible, e.g. resizing (see urxvt(7) for more info):

```
URxvt.keysym.M-C-3: command:\033[8;25;80t
URxvt.keysym.M-C-4: command:\033[8;48;110t
```

builtin:

The builtin action is the action that urxvt would execute if no key binding existed for the key combination. The obvious use is to undo the effect of existing bindings. The not so obvious use is to reinstate bindings when another binding overrides too many modifiers.

For example if you overwrite the `Insert` key you will disable urxvt's `Shift-Insert` mapping. To re-enable that, you can poke "holes" into the user-defined keymap using the `builtin:` replacement:

```
URxvt.keysym.Insert: <my insert key sequence>
URxvt.keysym.S-Insert: builtin:
```

The first line defines a mapping for `Insert` and *any* combination of modifiers. The second line re-establishes the default mapping for `Shift-Insert`.

builtin-string:

This action is mainly useful to restore string mappings for keys that have predefined actions in urxvt. The exact semantics are a bit difficult to explain - basically, this action will send the string to the application that would be sent if urxvt wouldn't have a built-in action for it.

An example might make it clearer: urxvt normally pastes the selection when you press `Shift-Insert`. With the following bindings, it would instead emit the (undocumented, but what applications running in the terminal might expect) sequence `ESC [ 2 $` instead:

```
URxvt.keysym.S-Insert: builtin-string:
URxvt.keysym.C-S-Insert: builtin:
```

The first line disables the paste functionality for that key combination, and the second reinstates the default behaviour for `Control-Shift-Insert`, which would otherwise be overridden.

Similarly, to let applications gain access to the `C-M-c` (copy to clipboard) and `C-M-v` (paste clipboard) key combination, you can do this:

```
URxvt.keysym.C-M-c: builtin-string:
URxvt.keysym.C-M-v: builtin-string:
```

EXTENSION:STRING

An action of this form invokes the action **STRING**, if any, provided by the urxvtperl(3) extension **EXTENSION**. The extension will be loaded automatically if necessary.

Not all extensions define actions, but popular extensions that do include the *selection* and *matcher* extensions (documented in their own manpages, urxvt-selection(1) and urxvt-matcher(1), respectively).

From the silly examples department, this will rot13-"encrypt" urxvt's selection when Alt-Control-c is pressed on typical PC keyboards:

```
URxvt.keysym.M-C-c: selection:rot13
```

perl:STRING *DEPRECATED*

This is a deprecated way of invoking commands provided by perl extensions. It is still supported, but should not be used anymore.

perl-ext-common: ***string***

perl-ext: ***string***

Comma-separated list(s) of perl extension scripts (default: `default`) to use in this terminal instance; option **-pe**.

Extension names can be prefixed with a `-` sign to remove them again, in case they had been specified earlier. This can be useful to selectively disable some extensions loaded by default, or specified via the `perl-ext-common` resource. For example, `default,-selection` will use all the default extensions except `selection`.

To prohibit autoloading of extensions, you can prefix them with `/`, which will make urxvt refuse to automatically load them (this can be overridden, however, by specifying the extension name again without a prefix, though). This does not prohibit extensions themselves loading other extensions. For example, `default,/background` will keep the `background` extension from being loaded when a background OSC sequence is received.

The default set includes the `selection`, `option-popup`, `selection-popup`, `readline`, `searchable-scrollback` and `confirm-paste` extensions, as well as any extensions which are mentioned in **keysym** resources.

Any extension such that a corresponding resource is given on the command line is automatically appended to **perl-ext**.

Each extension is looked up in the library directories, loaded if necessary, and bound to the current terminal instance. When the library search path contains multiple extension files of the same name, then the first one found will be used.

If both of these resources are the empty string, then the perl interpreter will not be initialized. The rationale for having two options is that **perl-ext-common** will be used for extensions that should be available to all instances, while **perl-ext** is used for specific instances.

perl-eval: ***string***

Perl code to be evaluated when all extensions have been registered. See the urxvtperl(3) manpage.

perl-lib: ***path***

Colon-separated list of additional directories that hold extension scripts. When looking for perl extensions, urxvt will first look in these directories, then in `$URXVT_PERL_LIB`, *$HOME/.urxvt/ext* and lastly in *(release docs will have LIBDIR here)/urxvt/perl/*.

See the urxvtperl(3) manpage.

selection.pattern-*idx*: ***perl-regex***

Additional selection patterns, see the urxvtperl(3) manpage for details.

selection-autotransform.*idx*: ***perl-transform***

Selection auto-transform patterns, see the urxvtperl(3) manpage for details.

searchable-scrollback: ***keysym*** ***DEPRECATED***

This resource is deprecated and will be removed. Use a **keysym** resource instead, e.g.:

```
URxvt.keysym.M-s: searchable-scrollback:start
```

url-launcher: ***string***

Specifies the program to be started with a URL argument. Used by the `selection-popup` and `matcher` perl extensions.

transient-for: ***windowid***

Compile *frills*: Sets the WM_TRANSIENT_FOR property to the given window id.

override-redirect: ***boolean***

Compile *frills*: Sets override-redirect for the terminal window, making it almost invisible to window managers; option **-override-redirect**.

iso14755: ***boolean***

Turn on/off ISO 14755 (default enabled).

iso14755_52: ***boolean***

Turn on/off ISO 14755 5.2 mode (default enabled).

THE SCROLLBAR

Lines of text that scroll off the top of the **urxvt** window (resource: **saveLines**) and can be scrolled back using the scrollbar or by keystrokes. The normal **urxvt** scrollbar has arrows and its behaviour is fairly intuitive. The **xterm-scrollbar** is without arrows and its behaviour mimics that of *xterm*

Scroll down with **Button1** (**xterm-scrollbar**) or **Shift-Next**. Scroll up with **Button3** (**xterm-scrollbar**) or **Shift-Prior**. Continuous scroll with **Button2**.

MOUSE REPORTING

To temporarily override mouse reporting, for either the scrollbar or the normal text selection/insertion, hold either the Shift or the Meta (Alt) key while performing the desired mouse action.

If mouse reporting mode is active, the normal scrollbar actions are disabled – on the assumption that we are using a fullscreen application. Instead, pressing Button1 and Button3 sends **ESC [6 ~** (Next) and **ESC [5 ~** (Prior), respectively. Similarly, clicking on the up and down arrows sends **ESC [A** (Up) and **ESC [B** (Down), respectively.

THE SELECTION: SELECTING AND PASTING TEXT

The behaviour of text selection and insertion/pasting mechanism is similar to *xterm*(1).

Selecting:

Left click at the beginning of the region, drag to the end of the region and release; Right click to extend the marked region; Left double-click to select a word; Left triple-click to select the entire logical line (which can span multiple screen lines), unless modified by resource **tripleclickwords**.

Starting a selection while pressing the **Meta** key (or **Meta+Ctrl** keys) (Compile: *frills*) will create a rectangular selection instead of a normal one. In this mode, every selected row becomes its own line in the selection, and trailing whitespace is visually underlined and removed from the selection.

Pasting:

Pressing and releasing the Middle mouse button in an **urxvt** window causes the value of the PRIMARY selection (or CLIPBOARD with the **Meta** modifier) to be inserted as if it had been typed on the keyboard.

Pressing **Shift-Insert** causes the value of the PRIMARY selection to be inserted too.

rxvt-unicode also provides the bindings **Ctrl-Meta-c** and <Ctrl-Meta-v> to interact with the CLIPBOARD selection. The first binding causes the value of the internal selection to be copied to the CLIPBOARD selection, while the second binding causes the value of the CLIPBOARD selection to be inserted.

CHANGING FONTS

Changing fonts (or font sizes, respectively) via the keypad is not yet supported in rxvt-unicode. Bug me if you need this.

You can, however, switch fonts at runtime using escape sequences, e.g.:

```
printf '\e]710;%s\007' "9x15bold,xft:Kochi Gothic"
```

You can use keyboard shortcuts, too:

```
URxvt.keysym.M-C-1: command:\033]710;suxuseuro\007\033]711;suxuseuro\007
URxvt.keysym.M-C-2: command:\033]710;9x15bold\007\033]711;9x15bold\007
```

rxvt-unicode will automatically re-apply these fonts to the output so far.

ISO 14755 SUPPORT

ISO 14755 is a standard for entering and viewing unicode characters and character codes using the keyboard. It consists of 4 parts. The first part is available if rxvt-unicode has been compiled with `--enable-frills`, the rest is available when rxvt-unicode was compiled with `--enable-iso14755`.

* **5.1: Basic method**

  This allows you to enter unicode characters using their hexcode.

  Start by pressing and holding both `Control` and `Shift`, then enter hex-digits (between one and six). Releasing `Control` and `Shift` will commit the character as if it were typed directly. While holding down `Control` and `Shift` you can also enter multiple characters by pressing `Space`, which will commit the current character and lets you start a new one.

  As an example of use, imagine a business card with a japanese e-mail address, which you cannot type. Fortunately, the card has the e-mail address printed as hexcodes, e.g. `671d 65e5`. You can enter this easily by pressing `Control` and `Shift`, followed by `6-7-1-D-SPACE-6-5-E-5`, followed by releasing the modifier keys.

* **5.2: Keyboard symbols entry method**

  This mode lets you input characters representing the keycap symbols of your keyboard, if representable in the current locale encoding.

  Start by pressing `Control` and `Shift` together, then releasing them. The next special key (cursor keys, home etc.) you enter will not invoke its usual function but instead will insert the corresponding keycap symbol. The symbol will only be entered when the key has been released, otherwise pressing e.g. `Shift` would enter the symbol for `ISO Level 2 Switch`, although your intention might have been to enter a reverse tab (Shift-Tab).

* **5.3: Screen-selection entry method**

  While this is implemented already (it's basically the selection mechanism), it could be extended by displaying a unicode character map.

* **5.4: Feedback method for identifying displayed characters for later input**

  This method lets you display the unicode character code associated with characters already displayed.

  You enter this mode by holding down `Control` and `Shift` together, then pressing and holding the left mouse button and moving around. The unicode hex code(s) (it might be a combining character) of the character under the pointer is displayed until you release `Control` and `Shift`.

  In addition to the hex codes it will display the font used to draw this character - due to implementation reasons, characters combined with combining characters, line drawing characters and unknown characters will always be drawn using the built-in support font.

With respect to conformance, rxvt-unicode is supposed to be compliant to both scenario A and B of ISO 14755, including part 5.2.

LOGIN STAMP

urxvt tries to write an entry into the *utmp*(5) file so that it can be seen via the *who(1)* command, and can accept messages. To allow this feature, **urxvt** may need to be installed setuid root on some systems or setgid to root or to some other group on others.

COLOURS AND GRAPHICS

In addition to the default foreground and background colours, **urxvt** can display up to 88/256 colours: 8 ANSI colours plus high-intensity (potentially bold/blink) versions of the same, and 72 (or 240 in 256 colour mode) colours arranged in an 4x4x4 (or 6x6x6) colour RGB cube plus a 8 (24) colour greyscale ramp.

urxvt supports direct 24-bit fg/bg RGB colour escapes `ESC [ 38 ; 2 ; R ; G ; Bm` / `ESC [ 48 ; 2; R ; G ; Bm` . However the number of 24-bit colours that can be used is limited: an internal 7x7x5 (256 colour mode) or 6x6x4 (88 colour mode) colour cube is used to index into the 24-bit colour space. When indexing collisions happen, the nearest old

colour in the cube will be adapted to the new 24-bit RGB colour. That means one cannot use many similar 24-bit colours. It's typically not a problem in common scenarios.

Here is a list of the ANSI colours with their names.

color0	(black)	= Black
color1	(red)	= Red3
color2	(green)	= Green3
color3	(yellow)	= Yellow3
color4	(blue)	= Blue3
color5	(magenta)	= Magenta3
color6	(cyan)	= Cyan3
color7	(white)	= AntiqueWhite
color8	(bright black)	= Grey25
color9	(bright red)	= Red
color10	(bright green)	= Green
color11	(bright yellow)	= Yellow
color12	(bright blue)	= Blue
color13	(bright magenta)	= Magenta
color14	(bright cyan)	= Cyan
color15	(bright white)	= White
foreground		= Black
background		= White

It is also possible to specify the colour values of **foreground**, **background**, **cursorColor**, **cursorColor2**, **colorBD**, **colorUL** as a number 0-15, as a convenient shorthand to reference the colour name of color0-color15.

The following text gives values for the standard 88 colour mode (and values for the 256 colour mode in parentheses).

The RGB cube uses indices 16..79 (16..231) using the following formulas:

```
index_88  = (r * 4 + g) * 4 + b + 16   # r, g, b = 0..3
index_256 = (r * 6 + g) * 6 + b + 16   # r, g, b = 0..5
```

The grayscale ramp uses indices 80..87 (232..239), from 10% to 90% in 10% steps (1/26 to 25/26 in 1/26 steps) - black and white are already part of the RGB cube.

Together, all those colours implement the 88 (256) colour xterm colours. Only the first 16 can be changed using resources currently, the rest can only be changed via command sequences ("escape codes").

Applications are advised to use terminfo or command sequences to discover number and RGB values of all colours (yes, you can query this...).

Note that **-rv** (**"reverseVideo: True"**) simulates reverse video by always swapping the foreground/background colours. This is in contrast to *xterm*(1) where the colours are only swapped if they have not otherwise been specified. For example,

```
urxvt -fg Black -bg White -rv
```

would yield White on Black, while on *xterm*(1) it would yield Black on White.

ALPHA CHANNEL SUPPORT

If Xft support has been compiled in and as long as Xft/Xrender/X don't get their act together, rxvt-unicode will do its own alpha channel management:

You can prefix any colour with an opaqueness percentage enclosed in brackets, i.e. `[percent]`, where `percent` is a decimal percentage (0-100) that specifies the opacity of the colour, where `0` is completely transparent and `100` is completely opaque. For example, `[50]red` is a half-transparent red, while `[95]#00ff00` is an almost opaque green. This is the recommended format to specify transparency values, and works with all ways to specify a colour.

For complete control, rxvt-unicode also supports `rgba:rrrr/gggg/bbbb/aaaa` (exactly four hex digits/component) colour specifications, where the additional `aaaa` component specifies opacity (alpha) values. The minimum value of `0000` is completely transparent, while `ffff` is completely opaque). The two example colours from earlier could also be specified as `rgba:ff00/0000/0000/8000` and `rgba:0000/ff00/0000/f332`.

You probably need to specify **"-depth 32"**, too, to force a visual with alpha channels, and have the luck that your X-server uses ARGB pixel layout, as X is far from just supporting ARGB visuals out of the box, and rxvt-unicode just fudges around.

For example, the following selects an almost completely transparent black background, and an almost opaque pink foreground:

```
urxvt -depth 32 -bg rgba:0000/0000/0000/4444 -fg "[80]pink"
```

When not using a background image, then the interpretation of the alpha channel is up to your compositing manager (most interpret it as transparency of course).

When using a background pixmap or pseudo-transparency, then the background colour will always behave as if it were completely transparent (so the background image shows instead), regardless of how it was specified, while other colours will either be transparent as specified (the background image will show through) on servers supporting the RENDER extension, or fully opaque on servers not supporting the RENDER EXTENSION.

Please note that due to bugs in Xft, specifying alpha values might result in garbage being displayed when the X-server does not support the RENDER extension.

ENVIRONMENT

urxvt sets and/or uses the following environment variables:

TERM

Normally set to `rxvt-unicode`, unless overwritten at configure time, via resources or on the command line.

COLORTERM

Either `rxvt`, `rxvt-xpm`, depending on whether urxvt was compiled with background image support, and optionally with the added extension `-mono` to indicate that rxvt-unicode runs on a monochrome screen.

COLORFGBG

Set to a string of the form `fg;bg` or `fg;xpm;bg`, where `fg` is the colour code used as default foreground/text colour (or the string `default` to indicate that the default-colour escape sequence is to be used), `bg` is the colour code used as default background colour (or the string `default`), and `xpm` is the string `default` if urxvt was compiled with background image support. Libraries like `ncurses` and `slang` can (and do) use this information to optimize screen output.

WINDOWID

Set to the (decimal) X Window ID of the urxvt window (the toplevel window, which usually has subwindows for the scrollbar, the terminal window and so on).

TERMINFO

Set to the terminfo directory iff urxvt was configured with `--with-terminfo=PATH`.

DISPLAY

Used by urxvt to connect to the display and set to the correct display in its child processes if `-display` isn't used to override. It defaults to `:0` if it doesn't exist.

SHELL

The shell to be used for command execution, defaults to `/bin/sh`.

RXVT_SOCKET [*sic*]

The unix domain socket path used by urxvtc(1) and urxvtd(1).

Default *$HOME/.urxvt/urxvtd-<nodename>*.

URXVT_PERL_LIB

Additional :-separated library search path for perl extensions. Will be searched after **-perl-lib** but before *~/.urxvt/ext* and the system library directory.

URXVT_PERL_VERBOSITY

See *urxvtperl*(3).

HOME

Used to locate the default directory for the unix domain socket for daemon communications and to locate various resource files (such as `.Xdefaults`)

XAPPLRESDIR

Directory where application-specific X resource files are located.

XENVIRONMENT

If set and accessible, gives the name of a X resource file to be loaded by urxvt.

FILES

/usr/lib/X11/rgb.txt

Colour names.

SEE ALSO

urxvt(7), urxvtc(1), urxvtd(1), urxvt-extensions(1), urxvtperl(3), xterm(1), sh(1), resize(1), X(1), pty(4), tty(4), utmp(5)

CURRENT PROJECT COORDINATOR

Project Coordinator

Marc A. Lehmann <rxvt-unicode@schmorp.de>.

http://software.schmorp.de/pkg/rxvt-unicode.html

AUTHORS

John Bovey

University of Kent, 1992, wrote the original Xvt.

Rob Nation <nation@rocket.sanders.lockheed.com>

very heavily modified Xvt and came up with Rxvt

Angelo Haritsis <ah@doc.ic.ac.uk>

wrote the Greek Keyboard Input (no longer in code)

mj olesen <olesen@me.QueensU.CA>

Wrote the menu system.

Project Coordinator (changes.txt 2.11 to 2.21)

Oezguer Kesim <kesim@math.fu-berlin.de>

Project Coordinator (changes.txt 2.21a to 2.4.5)

Geoff Wing <gcw@pobox.com>

Rewrote screen display and text selection routines.

Project Coordinator (changes.txt 2.4.6 - rxvt-unicode)

Marc Alexander Lehmann <rxvt-unicode@schmorp.de>

Forked rxvt-unicode, unicode support, rewrote almost all the code, perl extension, random hacks, numerous bugfixes and extensions.

Project Coordinator (Changes 1.0 -)

Emanuele Giaquinta <emanuele.giaquinta@gmail.com>

pty/utmp code rewrite, image code improvements, many random hacks and bugfixes.