Index

NAME

RXVT REFERENCE - FAQ, command sequences and other background information

SYNOPSIS

```
# set a new font set
printf '\33]50;%s\007' 9x15,xft:Kochi" Mincho"

# change the locale and tell rxvt-unicode about it
export LC_CTYPE=ja_JP.EUC-JP; printf "\33]701;$LC_CTYPE\007"

# set window title
printf '\33]2;%s\007' "new window title"
```

DESCRIPTION

This document contains the FAQ, the RXVT TECHNICAL REFERENCE documenting all escape sequences, and other background information.

The newest version of this document is also available on the World Wide Web at http://pod.tst.eu/http://cvs.schmorp.de/rxvt-unicode/doc/rxvt.7.pod.

The main manual page for urxvt itself is available at http://pod.tst.eu/http://cvs.schmorp.de/rxvt-unicode/doc/rxvt.1.pod.

RXVT-UNICODE/URXVT FREQUENTLY ASKED QUESTIONS

Meta, Features & Commandline Issues

My question isn't answered here, can I ask a human?

Before sending me mail, you could go to IRC: `irc.libera.chat`, channel `#rxvt-unicode` has some rxvt-unicode enthusiasts that might be interested in learning about new and exciting problems (but not FAQs :).

I use Gentoo, and I have a problem...

There are two big problems with Gentoo Linux: first, most if not all Gentoo systems are completely broken (missing or mismatched header files, broken compiler etc. are just the tip of the iceberg); secondly, it should be called Gentoo GNU/Linux.

For these reasons, it is impossible to support rxvt-unicode on Gentoo. Problems appearing on Gentoo systems will usually simply be ignored unless they can be reproduced on non-Gentoo systems.

Does it support tabs, can I have a tabbed rxvt-unicode?

Beginning with version 7.3, there is a perl extension that implements a simple tabbed terminal. It is installed by default, so any of these should give you tabs:

```
urxvt -pe tabbed

URxvt.perl-ext-common: default,tabbed
```

It will also work fine with tabbing functionality of many window managers or similar tabbing programs, and its embedding-features allow it to be embedded into other programs, as witnessed by *doc/rxvt-tabbed* or the upcoming `Gtk2::URxvt` perl module, which

features a tabbed urxvt (murxvt) terminal as an example embedding application.

How do I know which rxvt-unicode version I'm using?

The version number is displayed with the usage (-h). Also the escape sequence `ESC [ 8 n` sets the window title to the version number. When using the urxvtc client, the version displayed is that of the daemon.

Rxvt-unicode uses gobs of memory, how can I reduce that?

Rxvt-unicode tries to obey the rule of not charging you for something you don't use. One thing you should try is to configure out all settings that you don't need, for example, Xft support is a resource hog by design, when used. Compiling it out ensures that no Xft font will be loaded accidentally when rxvt-unicode tries to find a font for your characters.

Also, many people (me included) like large windows and even larger scrollback buffers: Without `--enable-unicode3`, rxvt-unicode will use 6 bytes per screen cell. For a 160x?? window this amounts to almost a kilobyte per line. A scrollback buffer of 10000 lines will then (if full) use 10 Megabytes of memory. With `--enable-unicode3` it gets worse, as rxvt-unicode then uses 8 bytes per screen cell.

How can I start urxvtd in a race-free way?

Try `urxvtd -f -o`, which tells urxvtd to open the display, create the listening socket and then fork.

How can I start urxvtd automatically when I run urxvtc?

If you want to start urxvtd automatically whenever you run urxvtc and the daemon isn't running yet, use this script:

```
#!/bin/sh
urxvtc "$@"
if [ $? -eq 2 ]; then
   urxvtd -q -o -f
   urxvtc "$@"
fi
```

This tries to create a new terminal, and if fails with exit status 2, meaning it couldn't connect to the daemon, it will start the daemon and re-run the command. Subsequent invocations of the script will re-use the existing daemon.

Another option is to use systemd socket-based activation (see systemd.socket(5)). Here is an example of a service unit file and of a socket unit file for the default socket path:

urxvtd.service

```
[Unit]
Description=urxvt terminal daemon
Requires=urxvtd.socket

[Service]
ExecStart=/usr/bin/urxvtd -o
```

urxvtd.socket

```
[Unit]
Description=urxvt terminal daemon socket

[Socket]
ListenStream=%h/.urxvt/urxvtd-%H

[Install]
WantedBy=sockets.target
```

How do I distinguish whether I'm running rxvt-unicode or a regular xterm? I need this to decide about setting colours etc.

The original rxvt and rxvt-unicode always export the variable "COLORTERM", so you can check and see if that is set. Note that several programs, JED, slrn, Midnight Commander automatically check this variable to decide whether or not to use colour.

How do I set the correct, full IP address for the DISPLAY variable?

If you've compiled rxvt-unicode with DISPLAY_IS_IP and have enabled insecure mode then it is possible to use the following shell script snippets to correctly set the display. If your version of rxvt-unicode wasn't also compiled with ESCZ_ANSWER (as assumed in these snippets) then the COLORTERM variable can be used to distinguish rxvt-unicode from a regular xterm.

Courtesy of Chuck Blake <cblake@BBN.COM> with the following shell script snippets:

```
# Bourne/Korn/POSIX family of shells:
[ ${TERM:-foo} = foo ] && TERM=xterm # assume an xterm if we don't know
if [ ${TERM:-foo} = xterm ]; then
   stty -icanon -echo min 0 time 15 # see if enhanced rxvt or not
   printf "\eZ"
   read term_id
   stty icanon echo
   if [ ""${term_id} = '^[[?1;2C' -a ${DISPLAY:-foo} = foo ]; then
      printf '\e[7n'        # query the rxvt we are in for the DISPLAY string
      read DISPLAY          # set it in our local shell
   fi
fi
```

How do I compile the manual pages on my own?

You need to have a recent version of perl installed as */usr/bin/perl*, one that comes with *pod2man*, *pod2text* and *pod2xhtml* (from *Pod::Xhtml*). Then go to the doc subdirectory and enter `make alldoc`.

Isn't rxvt-unicode supposed to be small? Don't all those features bloat?

I often get asked about this, and I think, no, they didn't cause extra bloat. If you compare a minimal rxvt and a minimal urxvt, you can see that the urxvt binary is larger (due to some encoding tables always being compiled in), but it actually uses less memory (RSS) after startup. Even with `--disable-everything`, this comparison is a bit unfair, as many features unique to urxvt (locale, encoding conversion, iso14755 etc.) are already in use in this mode.

```
  text    data     bss     drs     rss filename
 98398    1664      24   15695    1824 rxvt --disable-everything
188985    9048   66616   18222    1788 urxvt --disable-everything
```

When you `--enable-everything` (which *is* unfair, as this involves xft and full locale/XIM support which are quite bloaty inside libX11 and my libc), the two diverge, but not unreasonably so.

```
  text    data     bss     drs     rss filename
163431    2152      24   20123    2060 rxvt --enable-everything
1035683   49680   66648   29096    3680 urxvt --enable-everything
```

The very large size of the text section is explained by the east-asian encoding tables, which, if unused, take up disk space but nothing else and can be compiled out unless you rely on X11 core fonts that use those encodings. The BSS size comes from the 64k emergency buffer that my c++ compiler allocates (but of course doesn't use unless you are out of memory). Also, using an xft font instead of a core font immediately adds a few megabytes of RSS. Xft indeed is responsible for a lot of RSS even when not used.

Of course, due to every character using two or four bytes instead of one, a large scrollback buffer will ultimately make rxvt-unicode use more memory.

Compared to e.g. Eterm (5112k), aterm (3132k) and xterm (4680k), this still fares rather well. And compared to some monsters like gnome-terminal (21152k + extra 4204k in separate processes) or konsole (22200k + extra 43180k in daemons that stay around after exit, plus half a minute of startup time, including the hundreds of warnings it spits out), it fares extremely well *g*.

Why C++, isn't that unportable/bloated/uncool?

Is this a question? :) It comes up very often. The simple answer is: I had to write it, and C++ allowed me to write and maintain it in a fraction of the time and effort (which is a scarce resource for me). Put even shorter: It simply wouldn't exist without C++.

My personal stance on this is that C++ is less portable than C, but in the case of rxvt-unicode this hardly matters, as its portability limits are defined by things like X11, pseudo terminals, locale support and unix domain sockets, which are all less portable than C++ itself.

Regarding the bloat, see the above question: It's easy to write programs in C that use gobs of memory, and certainly possible to write programs in C++ that don't. C++ also often comes with large libraries, but this is not necessarily the case with GCC. Here is what rxvt links against on my system with a minimal config:

```
libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0x00002aaaaabc3000)
libc.so.6 => /lib/libc.so.6 (0x00002aaaaadde000)
libdl.so.2 => /lib/libdl.so.2 (0x00002aaaab01d000)
/lib64/ld-linux-x86-64.so.2 (0x00002aaaaaaab000)
```

And here is rxvt-unicode:

```
libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0x00002aaaaabc3000)
libgcc_s.so.1 => /lib/libgcc_s.so.1 (0x00002aaaaada2000)
libc.so.6 => /lib/libc.so.6 (0x00002aaaaaeb0000)
libdl.so.2 => /lib/libdl.so.2 (0x00002aaaab0ee000)
/lib64/ld-linux-x86-64.so.2 (0x00002aaaaaaab000)
```

No large bloated libraries (of course, none were linked in statically), except maybe libX11 :)

Rendering, Font & Look and Feel Issues

I can't get transparency working, what am I doing wrong?

First of all, transparency isn't officially supported in rxvt-unicode, so you are mostly on your own. Do not bug the author about it (but you may bug everybody else). Also, if you can't get it working consider it a rite of passage: ... and you failed.

Here are four ways to get transparency. **Do** read the manpage and option descriptions for the programs mentioned and rxvt-unicode. Really, do it!

1. Use pseudo-transparency:

```
Esetroot wallpaper.jpg
urxvt -tr -tint red -sh 40
```

That works. If you think it doesn't, you lack transparency and tinting support, or you are unable to read. This method requires that the background-setting program sets the _XROOTPMAP_ID or ESETROOT_PMAP_ID property. Compatible programs are Esetroot, hsetroot and feh.

2. Use a simple pixmap and emulate pseudo-transparency. This enables you to use effects other than tinting and shading: Just shade/tint/whatever your picture with gimp or any other tool:

```
convert wallpaper.jpg -blur 20x20 -modulate 30 background.jpg
urxvt -pixmap "background.jpg;:root"
```

That works. If you think it doesn't, you lack GDK-PixBuf support, or you are unable to read.

3. Use an ARGB visual:

```
urxvt -depth 32 -fg grey90 -bg rgba:0000/0000/4444/cccc
```

This requires XFT support, and the support of your X-server. If that doesn't work for you, find a working composite manager or window manager, both are required to support ARGB visuals for client windows.

4. Use xcompmgr and let it do the job:

```
xprop -frame -f _NET_WM_WINDOW_OPACITY 32c \
      -set _NET_WM_WINDOW_OPACITY 0xc0000000
```

Then click on a window you want to make transparent. Replace `0xc0000000` by other values to change the degree of opacity. If it doesn't work and your server crashes, you got to keep the pieces.

Why does rxvt-unicode sometimes leave pixel droppings?

Most fonts were not designed for terminal use, which means that character size varies a lot. A font that is otherwise fine for terminal use might contain some characters that are simply too wide. Rxvt-unicode will avoid these characters. For characters that are just "a bit" too wide a special "careful" rendering mode is used that redraws adjacent characters.

All of this requires that fonts do not lie about character sizes, however: Xft fonts often draw glyphs larger than their acclaimed bounding box, and rxvt-unicode has no way of detecting this (the correct way is to ask for the character bounding box, which unfortunately is wrong in these cases).

It's not clear (to me at least), whether this is a bug in Xft, freetype, or the respective font. If you encounter this problem you might try using the `-lsp` option to give the font more height. If that doesn't work, you might be forced to use a different font.

All of this is not a problem when using X11 core fonts, as their bounding box data is correct.

How can I keep rxvt-unicode from using reverse video so much?

First of all, make sure you are running with the right terminal settings (`TERM=rxvt-unicode`), which will get rid of most of these effects. Then make sure you have specified colours for italic and bold, as otherwise rxvt-unicode might use reverse video to simulate the effect:

```
URxvt.colorBD:  white
URxvt.colorIT:  green
```

Some programs assume totally weird colours (red instead of blue), how can I fix that?

For some unexplainable reason, some rare programs assume a very weird colour palette when confronted with a terminal with more than the standard 8 colours (rxvt-unicode supports 88). The right fix is, of course, to fix these programs not to assume non-ISO colours without very good reasons.

In the meantime, you can either edit your `rxvt-unicode` terminfo definition to only claim 8 colour support or use `TERM=rxvt`, which will fix colours but keep you from using other rxvt-unicode features.

Can I switch the fonts at runtime?

Yes, using an escape sequence. Try something like this, which has the same effect as using the `-fn` switch, and takes effect immediately:

```
printf '\33]50;%s\007' "9x15bold,xft:Kochi Gothic"
```

This is useful if you e.g. work primarily with japanese (and prefer a japanese font), but you have to switch to chinese temporarily, where japanese fonts would only be in your way.

You can think of this as a kind of manual ISO-2022 switching.

Why do italic characters look as if clipped?

Many fonts have difficulties with italic characters and hinting. For example, the otherwise very nicely hinted font `xft:Bitstream Vera Sans Mono` completely fails in its italic face. A workaround might be to enable freetype autohinting, i.e. like this:

```
URxvt.italicFont:        xft:Bitstream Vera Sans Mono:italic:autohint=true
URxvt.boldItalicFont:    xft:Bitstream Vera Sans Mono:bold:italic:autohint=true
```

Can I speed up Xft rendering somehow?

Yes, the most obvious way to speed it up is to avoid Xft entirely, as it is simply slow. If you still want Xft fonts you might try to disable antialiasing (by appending `:antialias=false`), which saves lots of memory and also speeds up rendering considerably.

Rxvt-unicode doesn't seem to anti-alias its fonts, what is wrong?

Rxvt-unicode will use whatever you specify as a font. If it needs to fall back to its default font search list it will prefer X11 core fonts, because they are small and fast, and then use Xft fonts. It has antialiasing disabled for most of them, because the author thinks they look best that way.

If you want antialiasing, you have to specify the fonts manually.

What's with this bold/blink stuff?

If no bold colour is set via `colorBD:`, bold will invert text using the standard foreground colour.

For the standard background colour, blinking will actually make the text blink when compiled with `--enable-text-blink`. Without `--enable-text-blink`, the blink attribute will be ignored.

On ANSI colours, bold/blink attributes are used to set high-intensity foreground/background colours.

color0-7 are the low-intensity colours.

color8-15 are the corresponding high-intensity colours.

I don't like the screen colours. How do I change them?

You can change the screen colours at run-time using *~/.Xdefaults* resources (or as long-options).

Here are values that are supposed to resemble a VGA screen, including the murky brown that passes for low-intensity yellow:

```
URxvt.color0:   #000000
URxvt.color1:   #A80000
URxvt.color2:   #00A800
URxvt.color3:   #A8A800
URxvt.color4:   #0000A8
URxvt.color5:   #A800A8
URxvt.color6:   #00A8A8
URxvt.color7:   #A8A8A8

URxvt.color8:   #000054
URxvt.color9:   #FF0054
URxvt.color10:  #00FF54
URxvt.color11:  #FFFF54
URxvt.color12:  #0000FF
URxvt.color13:  #FF00FF
URxvt.color14:  #00FFFF
URxvt.color15:  #FFFFFF
```

And here is a more complete set of non-standard colours.

```
URxvt.cursorColor:  #dc74d1
URxvt.pointerColor: #dc74d1
URxvt.background:   #0e0e0e
URxvt.foreground:   #4ad5e1
URxvt.color0:       #000000
URxvt.color8:       #8b8f93
URxvt.color1:       #dc74d1
URxvt.color9:       #dc74d1
URxvt.color2:       #0eb8c7
URxvt.color10:      #0eb8c7
URxvt.color3:       #dfe37e
URxvt.color11:      #dfe37e
URxvt.color5:       #9e88f0
URxvt.color13:      #9e88f0
URxvt.color6:       #73f7ff
URxvt.color14:      #73f7ff
URxvt.color7:       #e1dddd
```

```
URxvt.color15:      #e1dddd
```

They have been described (not by me) as "pretty girly".

Why do some characters look so much different than others?

See next entry.

How does rxvt-unicode choose fonts?

Most fonts do not contain the full range of Unicode, which is fine. Chances are that the font you (or the admin/package maintainer of your system/os) have specified does not cover all the characters you want to display.

rxvt-unicode makes a best-effort try at finding a replacement font. Often the result is fine, but sometimes the chosen font looks bad/ugly/wrong. Some fonts have totally strange characters that don't resemble the correct glyph at all, and rxvt-unicode lacks the artificial intelligence to detect that a specific glyph is wrong: it has to believe the font that the characters it claims to contain indeed look correct.

In that case, select a font of your taste and add it to the font list, e.g.:

```
urxvt -fn basefont,font2,font3...
```

When rxvt-unicode sees a character, it will first look at the base font. If the base font does not contain the character, it will go to the next font, and so on. Specifying your own fonts will also speed up this search and use less resources within rxvt-unicode and the X-server.

The only limitation is that none of the fonts may be larger than the base font, as the base font defines the terminal character cell size, which must be the same due to the way terminals work.

Why do some chinese characters look so different than others?

This is because there is a difference between script and language – rxvt-unicode does not know which language the text that is output is, as it only knows the unicode character codes. If rxvt-unicode first sees a japanese/chinese character, it might choose a japanese font for display. Subsequent japanese characters will use that font. Now, many chinese characters aren't represented in japanese fonts, so when the first non-japanese character comes up, rxvt-unicode will look for a chinese font – unfortunately at this point, it will still use the japanese font for chinese characters that are also in the japanese font.

The workaround is easy: just tag a chinese font at the end of your font list (see the previous question). The key is to view the font list as a preference list: If you expect more japanese, list a japanese font first. If you expect more chinese, put a chinese font first.

In the future it might be possible to switch language preferences at runtime (the internal data structure has no problem with using different fonts for the same character at the same time, but no interface for this has been designed yet).

Until then, you might get away with switching fonts at runtime (see Can I switch the fonts at runtime? later in this document).

How can I make mplayer display video correctly?

We are working on it, in the meantime, as a workaround, use something like:

```
urxvt -b 600 -geometry 20x1 -e sh -c 'mplayer -wid $WINDOWID file...'
```

Why is the cursor now blinking in emacs/vi/...?

This is likely caused by your editor/program's use of the `cvvis` terminfo capability. Emacs uses it by default, as well as some versions of vi and possibly other programs.

In emacs, you can switch that off by adding this to your `.emacs` file:

```
(setq visible-cursor nil)
```

For other programs, if they do not have an option, your have to remove the `cvvis` capability from the terminfo description.

When urxvt first added the blinking cursor option, it didn't add a `cvvis` capability, which served no purpose before. Version 9.21 introduced `cvvis` (and the ability to control blinking independent of cursor shape) for compatibility with other terminals, which traditionally use a blinking cursor for `cvvis`. This also reflects the intent of programs such as emacs, who expect `cvvis` to enable a blinking cursor.

Keyboard, Mouse & User Interaction

The new selection selects pieces that are too big, how can I select single words?

If you want to select e.g. alphanumeric words, you can use the following setting:

```
URxvt.selection.pattern-0: ([[:word:]]+)
```

If you click more than twice, the selection will be extended more and more.

To get a selection that is very similar to the old code, try this pattern:

```
URxvt.selection.pattern-0: ([^"&'()*,;<=>?@[\\\\]^`{|})]+)
```

Please also note that the *LeftClick Shift-LeftClick* combination also selects words like the old code.

I don't like the new selection/popups/hotkeys/perl, how do I change/disable it?

You can disable the perl extension completely by setting the **perl-ext-common** resource to the empty string, which also keeps rxvt-unicode from initialising perl, saving memory.

If you only want to disable specific features, you first have to identify which perl extension is responsible. For this, read the section **PREPACKAGED EXTENSIONS** in the urxvtperl(3) manpage. For example, to disable the **selection-popup** and **option-popup**, specify this **perl-ext-common** resource:

```
URxvt.perl-ext-common: default,-selection-popup,-option-popup
```

This will keep the default extensions, but disable the two popup extensions. Some extensions can also be configured, for example, scrollback search mode is triggered by **M-s**. You can move it to any other combination by adding a **keysym** resource that binds the desired combination to the `start` action of `searchable-scrollback` and another one that binds **M-s** to the `builtin:` action:

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

The cursor moves when selecting text in the current input line, how do I switch this off?

See next entry.

During rlogin/ssh/telnet/etc. sessions, clicking near the cursor outputs strange escape sequences, how do I fix this?

These are caused by the `readline` perl extension. Under normal circumstances, it will move your cursor around when you click into the line that contains it. It tries hard not to do this at the wrong moment, but when running a program that doesn't parse cursor movements or in some cases during rlogin sessions, it fails to detect this properly.

You can permanently switch this feature off by disabling the `readline` extension:

```
URxvt.perl-ext-common: default,-readline
```

My numeric keypad acts weird and generates differing output?

Some Debian GNU/Linux users seem to have this problem, although no specific details were reported so far. It is possible that this is caused by the wrong `TERM` setting, although the details of whether and how this can happen are unknown, as `TERM=rxvt` should offer a compatible keymap. See the answer to the previous question, and please report if that helped.

My Compose (Multi_key) key is no longer working.

The most common causes for this are that either your locale is not set correctly, or you specified a **preeditType** that is not supported by your input method. For example, if you specified **OverTheSpot** and your input method (e.g. the default input method handling Compose keys) does not support this (for instance because it is not visual), then rxvt-unicode will continue without an input method.

In this case either do not specify a **preeditType** or specify more than one pre-edit style, such as **OverTheSpot,Root,None**.

If it still doesn't work, then maybe your input method doesn't support compose sequences - to fall back to the built-in one, make sure you don't specify an input method via `-im` or `XMODIFIERS`.

I cannot type `Ctrl-Shift-2` to get an ASCII NUL character due to ISO 14755

Either try `Ctrl-2` alone (it often is mapped to ASCII NUL even on international keyboards) or simply use ISO 14755 support to your advantage, typing <Ctrl-Shift-O> to get a ASCII NUL. This works for other codes, too, such as `Ctrl-Shift-1-d` to type the default telnet escape character and so on.

Mouse cut/paste suddenly no longer works.

Make sure that mouse reporting is actually turned off since killing some editors prematurely may leave it active. I've heard that tcsh may use mouse reporting unless it is otherwise specified. A quick check is to see if cut/paste works when the Alt or Shift keys are pressed.

What's with the strange Backspace/Delete key behaviour?

Assuming that the physical Backspace key corresponds to the Backspace keysym (not likely for Linux ... see the following question) there are two standard values that can be used for Backspace: `^H` and `^?`.

Historically, either value is correct, but rxvt-unicode adopts the debian policy of using `^?` when unsure, because it's the one and only correct choice :).

It is possible to toggle between `^H` and `^?` with the DECBKM private mode:

```
# use Backspace = ^H
$ stty erase ^H
$ printf "\e[?67h"

# use Backspace = ^?
$ stty erase ^?
$ printf "\e[?67l"
```

This helps satisfy some of the Backspace discrepancies that occur, but if you use Backspace = `^H`, make sure that the termcap/terminfo value properly reflects that.

The Delete key is a another casualty of the ill-defined Backspace problem. To avoid confusion between the Backspace and Delete keys, the Delete key has been assigned an escape sequence to match the vt100 for Execute (`ESC [ 3 ~`) and is in the supplied termcap/terminfo.

Some other Backspace problems:

some editors use termcap/terminfo, some editors (vim I'm told) expect Backspace = ^H, GNU Emacs (and Emacs-like editors) use ^H for help.

Perhaps someday this will all be resolved in a consistent manner.

I don't like the key-bindings. How do I change them?

There are some compile-time selections available via configure. Unless you have run "configure" with the `--disable-resources` option you can use the `keysym' resource to alter the keystrings associated with keysyms.

Here's an example for a URxvt session started using `urxvt -name URxvt`

```
URxvt.keysym.Prior:         \033[5~
URxvt.keysym.Next:          \033[6~
URxvt.keysym.Home:          \033[7~
URxvt.keysym.End:           \033[8~
URxvt.keysym.Up:            \033[A
URxvt.keysym.Down:          \033[B
URxvt.keysym.Right:         \033[C
URxvt.keysym.Left:          \033[D
```

See some more examples in the documentation for the **keysym** resource.

I'm using keyboard model XXX that has extra Prior/Next/Insert keys. How do I make use of them? For example, the Sun Keyboard type 4 has the following map

```
KP_Insert == Insert
F22 == Print
F27 == Home
F29 == Prior
F33 == End
F35 == Next
```

Rather than have rxvt-unicode try to accommodate all the various possible keyboard mappings, it is better to use `xmodmap' to remap the keys as required for your particular machine.

Terminal Configuration

Can I see a typical configuration?

The default configuration tries to be xterm-like, which I don't like that much, but it's least surprise to regular users.

As a rxvt or rxvt-unicode user, you are practically supposed to invest time into customising your terminal. To get you started, here is the author's .Xdefaults entries, with comments on what they do. It's certainly not *typical*, but what's typical...

```
URxvt.cutchars: "()*,<>[]{}|'
URxvt.print-pipe: cat >/some/path
```

These are just for testing stuff.

```
URxvt.imLocale: ja_JP.UTF-8
URxvt.preeditType: OnTheSpot,None
```

This tells rxvt-unicode to use a special locale when communicating with the X Input Method, and also tells it to only use the OnTheSpot pre-edit type, which requires the `xim-onthespot` perl extension but rewards me with correct-looking fonts.

```
URxvt.perl-lib: /root/lib/urxvt
URxvt.perl-ext-common: default,selection-autotransform,selection-pastebin,xim-onthespot,remote-clipboard
URxvt.selection.pattern-0: ( at .*? line \\d+)
URxvt.selection.pattern-1: ^(/[^:]+):\
URxvt.selection-autotransform.0: s/^([^:[:space:]]+):(\\d+):?$/:e \\Q$1\\E\\x0d:$2\\x0d/
URxvt.selection-autotransform.1: s/^ at (.*?) line (\\d+)$/:e \\Q$1\\E\\x0d:$2\\x0d/
```

This is my perl configuration. The first two set the perl library directory and also tells urxvt to use a large number of extensions. I develop for myself mostly, so I actually use most of the extensions I write.

The selection stuff mainly makes the selection perl-error-message aware and tells it to convert perl error messages into vi-commands to load the relevant file and go to the error line number.

```
URxvt.scrollstyle:      plain
URxvt.secondaryScroll:  true
```

As the documentation says: plain is the preferred scrollbar for the author. The `secondaryScroll` configures urxvt to scroll in full-screen apps, like screen, so lines scrolled out of screen end up in urxvt's scrollback buffer.

```
URxvt.background:       #000000
URxvt.foreground:       gray90
URxvt.color7:           gray90
URxvt.colorBD:          #ffffff
URxvt.cursorColor:      #e0e080
URxvt.throughColor:     #8080f0
URxvt.highlightColor:   #f0f0f0
```

Some colours. Not sure which ones are being used or even non-defaults, but these are in my .Xdefaults. Most notably, they set foreground/background to light gray/black, and also make sure that the colour 7 matches the default foreground colour.

```
URxvt.underlineColor:   yellow
```

Another colour, makes underline lines look different. Sometimes hurts, but is mostly a nice effect.

```
URxvt.geometry:         154x36
URxvt.loginShell:       false
URxvt.meta:             ignore
URxvt.utmpInhibit:      true
```

Uh, well, should be mostly self-explanatory. By specifying some defaults manually, I can quickly switch them for testing.

```
URxvt.saveLines:        8192
```

A large scrollback buffer is essential. Really.

```
URxvt.mapAlert:         true
```

The only case I use it is for my IRC window, which I like to keep iconified till people msg me (which beeps).

```
URxvt.visualBell:       true
```

The audible bell is often annoying, especially when in a crowd.

```
URxvt.insecure:         true
```

Please don't hack my mutt! Ooops...

```
URxvt.pastableTabs:     false
```

I once thought this is a great idea.

```
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, \
                        xft:Bitstream Vera Sans Mono:autohint=true, \
                        xft:Code2000:antialias=false
urxvt.boldFont:         -xos4-terminus-bold-r-normal--14-140-72-72-c-80-iso8859-15
urxvt.italicFont:       xft:Bitstream Vera Sans Mono:italic:autohint=true
urxvt.boldItalicFont:   xft:Bitstream Vera Sans Mono:bold:italic:autohint=true
```

I wrote rxvt-unicode to be able to specify fonts exactly. So don't be overwhelmed. A special note: the `9x15bold` mentioned above is actually the version from XFree-3.3, as XFree-4 replaced it by a totally different font (different glyphs for `;` and many other harmless characters), while the second font is actually the `9x15bold` from XFree4/XOrg. The bold version has less chars than the medium version, so I use it for rare characters, too. When editing sources with vim, I use italic for comments and other stuff, which looks quite good with Bitstream Vera anti-aliased.

Terminus is a quite bad font (many very wrong glyphs), but for most of my purposes, it works, and gives a different look, as my normal (Non-bold) font is already bold, and I want to see a difference between bold and normal fonts.

Please note that I used the `urxvt` instance name and not the `URxvt` class name. That is because I use different configs for different purposes, for example, my IRC window is started with `-name IRC`, and uses these defaults:

```
IRC*title:              IRC
IRC*geometry:           87x12+535+542
IRC*saveLines:          0
IRC*mapAlert:           true
IRC*font:               suxuseuro
IRC*boldFont:           suxuseuro
IRC*colorBD:            white
IRC*keysym.M-C-1:       command:\033]710;suxuseuro\007\033]711;suxuseuro\007
IRC*keysym.M-C-2:       command:\033]710;9x15bold\007\033]711;9x15bold\007
```

`Alt-Ctrl-1` and `Alt-Ctrl-2` switch between two different font sizes. `suxuseuro` allows me to keep an eye (and actually read) stuff while keeping a very small window. If somebody pastes something complicated (e.g. japanese), I temporarily switch to a larger font.

The above is all in my `.Xdefaults` (I don't use `.Xresources` nor `xrdb`). I also have some resources in a separate `.Xdefaults-hostname` file for different hosts, for example, on my main desktop, I use:

```
URxvt.keysym.C-M-q: command:\033[3;5;5t
URxvt.keysym.C-M-y: command:\033[3;5;606t
URxvt.keysym.C-M-e: command:\033[3;1605;5t
URxvt.keysym.C-M-c: command:\033[3;1605;606t
URxvt.keysym.C-M-p: perl:test
```

The first for keysym definitions allow me to quickly bring some windows in the layout I like most. Ion users might start laughing but will stop immediately when I tell them that I use my own Fvwm2 module for much the same effect as Ion provides, and I only very rarely use the above key combinations :->

Why doesn't rxvt-unicode read my resources?

Well, why, indeed? It does, in a way very similar to other X applications. Most importantly, this means that if you or your OS loads resources into the X display (the right way to do it), rxvt-unicode will ignore any resource files in your home directory. It will only read *$HOME/.Xdefaults* when no resources are attached to the display.

If you have or use an *$HOME/.Xresources* file, chances are that resources are loaded into your X-server. In this case, you have to re-login after every change (or run *xrdb -merge $HOME/.Xresources*).

Also consider the form resources have to use:

```
URxvt.resource: value
```

If you want to use another form (there are lots of different ways of specifying resources), make sure you understand whether and why it works. If unsure, use the form above.

When I log-in to another system it tells me about missing terminfo data?

The terminal description used by rxvt-unicode is not as widely available as that for xterm, or even rxvt (for which the same problem often arises).

The correct solution for this problem is to install the terminfo, this can be done by simply installing rxvt-unicode on the remote system as well (in case you have a nice package manager ready), or you can install the terminfo database manually like this (with ncurses infocmp. works as user and root):

```
REMOTE=remotesystem.domain
infocmp rxvt-unicode | ssh $REMOTE "mkdir -p .terminfo && cat >/tmp/ti && tic /tmp/ti"
```

One some systems you might need to set `$TERMINFO` to the full path of *$HOME/.terminfo* for this to work.

If you cannot or do not want to do this, then you can simply set `TERM=rxvt` or even `TERM=xterm`, and live with the small number of problems arising, which includes wrong keymapping, less and different colours and some refresh errors in fullscreen applications. It's a nice quick-and-dirty workaround for rare cases, though.

If you always want to do this (and are fine with the consequences) you can either recompile rxvt-unicode with the desired TERM value or use a resource to set it:

```
URxvt.termName: rxvt
```

If you don't plan to use **rxvt** (quite common...) you could also replace the rxvt terminfo file with the rxvt-unicode one and use `TERM=rxvt`.

nano fails with "Error opening terminal: rxvt-unicode"

This exceptionally confusing and useless error message is printed by nano when it can't find the terminfo database. Nothing is wrong with your terminal, read the previous answer for a solution.

`tic` outputs some error when compiling the terminfo entry.

Most likely it's the empty definition for `enacs=`. Just replace it by `enacs=\E[0@` and try again.

`bash`'s readline does not work correctly under urxvt.

See next entry.

I need a termcap file entry.

One reason you might want this is that some distributions or operating systems still compile some programs using the long-obsoleted termcap library (Fedora's bash is one example) and rely on a termcap entry for `rxvt-unicode`.

You could use rxvt's termcap entry with reasonable results in many cases. You can also create a termcap entry by using terminfo's infocmp program like this:

```
infocmp -C rxvt-unicode
```

Or you could use the termcap entry in doc/etc/rxvt-unicode.termcap, generated by the command above.

Why does `ls` no longer have coloured output?

The `ls` in the GNU coreutils unfortunately doesn't use terminfo to decide whether a terminal has colour, but uses its own configuration file. Needless to say, `rxvt-unicode` is not in its default file (among with most other terminals supporting colour). Either add:

```
TERM rxvt-unicode
```

to `/etc/DIR_COLORS` or simply add:

```
alias ls='ls --color=auto'
```

to your `.profile` or `.bashrc`.

Why doesn't vim/emacs etc. use the 88 colour mode?

See next entry.

Why doesn't vim/emacs etc. make use of italic?

See next entry.

Why are the secondary screen-related options not working properly?

Make sure you are using `TERM=rxvt-unicode`. Some pre-packaged distributions break rxvt-unicode by setting `TERM` to `rxvt`, which doesn't have these extra features. Unfortunately, some of these furthermore fail to even install the `rxvt-unicode` terminfo file, so you will need to install it on your own (See the question **When I log-in to another system it tells me about missing terminfo data?** on how to do this).

Encoding / Locale / Input Method Issues

Rxvt-unicode does not seem to understand the selected encoding?

See next entry.

Unicode does not seem to work?

If you encounter strange problems like typing an accented character but getting two unrelated other characters or similar, or if program output is subtly garbled, then you should check your locale settings.

Rxvt-unicode must be started with the same `LC_CTYPE` setting as the programs running in it. Often rxvt-unicode is started in the `C` locale, while the login script running within the rxvt-unicode window changes the locale to something else, e.g. `en_GB.UTF-8`. Needless to say, this is not going to work, and is the most common cause for problems.

The best thing is to fix your startup environment, as you will likely run into other problems. If nothing works you can try this in your .profile.

```
printf '\33]701;%s\007' "$LC_CTYPE"   # $LANG or $LC_ALL are worth a try, too
```

If this doesn't work, then maybe you use a `LC_CTYPE` specification not supported on your systems. Some systems have a `locale` command which displays this (also, `perl -e0` can be used to check locale settings, as it will complain loudly if it cannot set the locale). If it displays something like:

```
locale: Cannot set LC_CTYPE to default locale: ...
```

Then the locale you specified is not supported on your system.

If nothing works and you are sure that everything is set correctly then you will need to remember a little known fact: Some programs just don't support locales :(

How does rxvt-unicode determine the encoding to use?

See next entry.

Is there an option to switch encodings?

Unlike some other terminals, rxvt-unicode has no encoding switch, and no specific "utf-8" mode, such as xterm. In fact, it doesn't even know about UTF-8 or any other encodings with respect to terminal I/O.

The reasons is that there exists a perfectly fine mechanism for selecting the encoding, doing I/O and (most important) communicating this to all applications so everybody agrees on character properties such as width and code number. This mechanism is the *locale*. Applications not using that info will have problems (for example, `xterm` gets the width of characters wrong as it uses its own, locale-independent table under all locales).

Rxvt-unicode uses the `LC_CTYPE` locale category to select encoding. All programs doing the same (that is, most) will automatically agree in the interpretation of characters.

Unfortunately, there is no system-independent way to select locales, nor is there a standard on how locale specifiers will look like.

On most systems, the content of the `LC_CTYPE` environment variable contains an arbitrary string which corresponds to an already-installed locale. Common names for locales are `en_US.UTF-8`, `de_DE.ISO-8859-15`, `ja_JP.EUC-JP`, i.e. `language_country.encoding`, but other forms (i.e. `de` or `german`) are also common.

Rxvt-unicode ignores all other locale categories, and except for the encoding, ignores country or language-specific settings, i.e. `de_DE.UTF-8` and `ja_JP.UTF-8` are the normally same to rxvt-unicode.

If you want to use a specific encoding you have to make sure you start rxvt-unicode with the correct `LC_CTYPE` category.

Can I switch locales at runtime?

Yes, using an escape sequence. Try something like this, which sets rxvt-unicode's idea of `LC_CTYPE`.

```
printf '\33]701;%s\007' ja_JP.SJIS
```

See also the previous answer.

Sometimes this capability is rather handy when you want to work in one locale (e.g. `de_DE.UTF-8`) but some programs don't support it (e.g. UTF-8). For example, I use this script to start `xjdic`, which first switches to a locale supported by xjdic and back later:

```
printf '\33]701;%s\007' ja_JP.SJIS
xjdic -js
printf '\33]701;%s\007' de_DE.UTF-8
```

You can also use xterm's `luit` program, which usually works fine, except for some locales where character width differs between program- and rxvt-unicode-locales.

I have problems getting my input method working.

Try a search engine, as this is slightly different for every input method server.

Here is a checklist:

- **Make sure your locale *and* the imLocale are supported on your OS.**

  Try `locale -a` or check the documentation for your OS.

- **Make sure your locale or imLocale matches a locale supported by your XIM.**

  For example, **kinput2** does not support UTF-8 locales, you should use `ja_JP.EUC-JP` or equivalent.

- **Make sure your XIM server is actually running.**
- **Make sure the `XMODIFIERS` environment variable is set correctly when *starting* rxvt-unicode.**

  When you want to use e.g. **kinput2**, it must be set to `@im=kinput2`. For **scim**, use `@im=SCIM`. You can see what input method servers are running with this command:

  ```
  xprop -root XIM_SERVERS
  ```

My input method wants <some encoding> but I want UTF-8, what can I do?

You can specify separate locales for the input method and the rest of the terminal, using the resource `imlocale`:

```
URxvt.imlocale: ja_JP.EUC-JP
```

Now you can start your terminal with `LC_CTYPE=ja_JP.UTF-8` and still use your input method. Please note, however, that, depending on your Xlib version, you may not be able to input characters outside `EUC-JP` in a normal way then, as your input method

limits you.

Rxvt-unicode crashes when the X Input Method changes or exits.

Unfortunately, this is unavoidable, as the XIM protocol is racy by design. Applications can avoid some crashes at the expense of memory leaks, and Input Methods can avoid some crashes by careful ordering at exit time. **kinput2** (and derived input methods) generally succeeds, while **SCIM** (or similar input methods) fails. In the end, however, crashes cannot be completely avoided even if both sides cooperate.

So the only workaround is not to kill your Input Method Servers.

Operating Systems / Package Maintaining

I am maintaining rxvt-unicode for distribution/OS XXX, any recommendation?

You should build one binary with the default options. *configure* now enables most useful options, and the trend goes to making them runtime-switchable, too, so there is usually no drawback to enabling them, except higher disk and possibly memory usage. The perl interpreter should be enabled, as important functionality (menus, selection, likely more in the future) depends on it.

You should not overwrite the `perl-ext-common` and `perl-ext` resources system-wide (except maybe with `defaults`). This will result in useful behaviour. If your distribution aims at low memory, add an empty `perl-ext-common` resource to the app-defaults file. This will keep the perl interpreter disabled until the user enables it.

If you can/want build more binaries, I recommend building a minimal one with `--disable-everything` (very useful) and a maximal one with `--enable-everything` (less useful, it will be very big due to a lot of encodings built-in that increase download times and are rarely used).

I need to make it setuid/setgid to support utmp/ptys on my OS, is this safe?

It should be, starting with release 7.1. You are encouraged to properly install urxvt with privileges necessary for your OS now.

When rxvt-unicode detects that it runs setuid or setgid, it will fork into a helper process for privileged operations (pty handling on some systems, utmp/wtmp/lastlog handling on others) and drop privileges immediately. This is much safer than most other terminals that keep privileges while running (but is more relevant to urxvt, as it contains things as perl interpreters, which might be "helpful" to attackers).

This forking is done as the very first within main(), which is very early and reduces possible bugs to initialisation code run before main(), or things like the dynamic loader of your system, which should result in very little risk.

I am on FreeBSD and rxvt-unicode does not seem to work at all.

Rxvt-unicode requires the symbol `__STDC_ISO_10646__` to be defined in your compile environment, or an implementation that implements it, whether it defines the symbol or not. `__STDC_ISO_10646__` requires that **wchar_t** is represented as unicode.

As you might have guessed, FreeBSD does neither define this symbol nor does it support it. Instead, it uses its own internal representation of **wchar_t**. This is, of course, completely fine with respect to standards.

However, that means rxvt-unicode only works in `POSIX`, `ISO-8859-1` and `UTF-8` locales under FreeBSD (which all use Unicode as **wchar_t**).

`__STDC_ISO_10646__` is the only sane way to support multi-language apps in an OS, as using a locale-dependent (and non-standardized) representation of **wchar_t** makes it impossible to convert between **wchar_t** (as used by X11 and your applications) and any other encoding without implementing OS-specific-wrappers for each and every locale. There simply are no APIs to convert **wchar_t** into anything except the current locale encoding.

Some applications (such as the formidable **mlterm**) work around this by carrying their own replacement functions for character set handling with them, and either implementing OS-dependent hacks or doing multiple conversions (which is slow and unreliable in case the OS implements encodings slightly different than the terminal emulator).

The rxvt-unicode author insists that the right way to fix this is in the system libraries once and for all, instead of forcing every app to carry complete replacements for them :)

How can I use rxvt-unicode under cygwin?

rxvt-unicode should compile and run out of the box on cygwin, using the X11 libraries that come with cygwin. libW11 emulation is no longer supported (and makes no sense, either, as it only supported a single font). I recommend starting the X-server in `-multiwindow` or `-rootless` mode instead, which will result in similar look&feel as the old libW11 emulation.

At the time of this writing, cygwin didn't seem to support any multi-byte encodings (you might try `LC_CTYPE=C-UTF-8`), so you are likely limited to 8-bit encodings.

Character widths are not correct.

urxvt uses the system wcwidth function to know the information about the width of characters, so on systems with incorrect locale data you will likely get bad results. Two notorious examples are Solaris 9, where single-width characters like U+2514 are reported as double-width, and Darwin 8, where combining chars are reported having width 1.

The solution is to upgrade your system or switch to a better one. A possibly working workaround is to use a wcwidth implementation like

http://www.cl.cam.ac.uk/~mgk25/ucs/wcwidth.c

RXVT-UNICODE TECHNICAL REFERENCE

The rest of this document describes various technical aspects of **rxvt-unicode**. First the description of supported command sequences, followed by pixmap support and last by a description of all features selectable at `configure` time.

When some functionality is marked as (insecure mode), then it requires insecure mode to be enabled to work fully, e.g. by using the **insecure** resource or command line switch. As that name implies, a terminal running in insecure mode might not be secure against attackers that can output arbitrary sequences to the terminal.

Definitions

`c`

The literal character c (potentially a multi-byte character).

`C`

A single (required) character.

`Ps`

A single (usually optional) numeric parameter, composed of one or more digits.

`Pm`

A multiple numeric parameter composed of any number of single numeric parameters, separated by `;` character(s).

`Pt`

A text parameter composed of printable characters.

Values

`ENQ`

Enquiry (Ctrl-E) = Send Device Attributes (DA) request attributes from terminal. See **`ESC [ Ps c`**.

`BEL`

Bell (Ctrl-G)

`BS`

Backspace (Ctrl-H)

TAB

Horizontal Tab (HT) (Ctrl-I)

LF

Line Feed or New Line (NL) (Ctrl-J)

VT

Vertical Tab (Ctrl-K) same as **LF**

FF

Form Feed or New Page (NP) (Ctrl-L) same as **LF**

CR

Carriage Return (Ctrl-M)

SO

Shift Out (Ctrl-N), invokes the G1 character set. Switch to Alternate Character Set

SI

Shift In (Ctrl-O), invokes the G0 character set (the default). Switch to Standard Character Set

SP

Space Character

Escape Sequences

ESC # 8

DEC Screen Alignment Test (DECALN)

ESC 7

Save Cursor (SC)

ESC 8

Restore Cursor

ESC =

Application Keypad (SMKX). See also next sequence.

ESC >

Normal Keypad (RMKX)

Note: numbers or control functions are generated by the numeric keypad in normal or application mode, respectively (see Key Codes).

ESC D

Index (IND)

ESC E

Next Line (NEL)

ESC H

Tab Set (HTS)

ESC M

Reverse Index (RI)

ESC N

Single Shift Select of G2 Character Set (SS2): affects next character only *unimplemented*

ESC O

Single Shift Select of G3 Character Set (SS3): affects next character only *unimplemented*

ESC Z

Obsolete form of returns: **ESC [? 1 ; 2 C** *rxvt-unicode compile-time option*

ESC c

Full reset (RIS)

ESC n

Invoke the G2 Character Set (LS2)

ESC o

Invoke the G3 Character Set (LS3)

ESC (C

Designate G0 Character Set (ISO 2022), see below for values of `C`.

ESC) C

Designate G1 Character Set (ISO 2022), see below for values of `C`.

ESC * C

Designate G2 Character Set (ISO 2022), see below for values of `C`.

ESC + C

Designate G3 Character Set (ISO 2022), see below for values of `C`.

ESC $ C

Designate Kanji Character Set

Where **C** is one of:

C = 0	DEC Special Character and Line Drawing Set
C = A	United Kingdom (UK)
C = B	United States (USASCII)
C = <	Multinational character set unimplemented
C = 5	Finnish character set unimplemented
C = C	Finnish character set unimplemented
C = K	German character set unimplemented

CSI (Command Sequence Introducer) Sequences

ESC [Ps @

Insert **Ps** (Blank) Character(s) [default: 1] (ICH)

ESC [Ps A

Cursor Up **Ps** Times [default: 1] (CUU)

ESC [Ps B

Cursor Down **Ps** Times [default: 1] (CUD)

ESC [Ps C

Cursor Forward **Ps** Times [default: 1] (CUF)

ESC [Ps D

Cursor Backward **Ps** Times [default: 1] (CUB)

ESC [Ps E

Cursor Down **Ps** Times [default: 1] and to first column

ESC [Ps F

Cursor Up **Ps** Times [default: 1] and to first column

ESC [Ps G

Cursor to Column **Ps** (HPA)

ESC [Ps;Ps H

Cursor Position [row;column] [default: 1;1] (CUP)

ESC [Ps I

Move forward **Ps** tab stops [default: 1]

ESC [Ps J

Erase in Display (ED)

Ps = 0	Clear Right and Below (default)
Ps = 1	Clear Left and Above
Ps = 2	Clear All

ESC [Ps K

Erase in Line (EL)

Ps = 0	Clear to Right (default)
Ps = 1	Clear to Left
Ps = 2	Clear All
Ps = 3	Like Ps = 0, but is ignored when wrapped (urxvt extension)

ESC [Ps L

Insert **Ps** Line(s) [default: 1] (IL)

ESC [Ps M

Delete **Ps** Line(s) [default: 1] (DL)

ESC [Ps P

Delete **Ps** Character(s) [default: 1] (DCH)

ESC [Ps;Ps;Ps;Ps;Ps T

Initiate . *unimplemented* Parameters are [func;startx;starty;firstrow;lastrow].

ESC [Ps W

Tabulator functions

Ps = 0	Tab Set (HTS)
Ps = 2	Tab Clear (TBC), Clear Current Column (default)
Ps = 5	Tab Clear (TBC), Clear All

ESC [Ps X

Erase **Ps** Character(s) [default: 1] (ECH)

ESC [Ps Z

Move backward **Ps** [default: 1] tab stops

ESC [Ps '

See **ESC [Ps G**

ESC [Ps a

See **ESC [Ps C**

ESC [Ps c

Send Device Attributes (DA) **Ps = 0** (or omitted): request attributes from terminal returns: **ESC [? 1 ; 2 c** (``I am a VT100 with Advanced Video Option'')

ESC [Ps d

Cursor to Line **Ps** (VPA)

ESC [Ps e

See **ESC [Ps A**

ESC [Ps;Ps f

Horizontal and Vertical Position [row;column] (HVP) [default: 1;1]

ESC [Ps g

Tab Clear (TBC)

Ps = 0	Clear Current Column (default)
Ps = 3	Clear All (TBC)

ESC [Pm h

Set Mode (SM). See **ESC [Pm l** sequence for description of `Pm`.

ESC [Ps i

Printing. See also the `print-pipe` resource.

Ps = 0	print screen (MC0)
Ps = 4	disable transparent print mode (MC4)
Ps = 5	enable transparent print mode (MC5)

ESC [Pm l

Reset Mode (RM)

Ps = 4

h	Insert Mode (SMIR)
l	Replace Mode (RMIR)

Ps = 20 (partially implemented)

h	Automatic Newline (LNM)
l	Normal Linefeed (LNM)

ESC [Pm m

Character Attributes (SGR)

Pm = 0	Normal (default)
Pm = 1 / 21	On / Off Bold (bright fg)
Pm = 3 / 23	On / Off Italic
Pm = 4 / 24	On / Off Underline
Pm = 5 / 25	On / Off Slow Blink (bright bg)
Pm = 6 / 26	On / Off Rapid Blink (bright bg)
Pm = 7 / 27	On / Off Inverse
Pm = 8 / 27	On / Off Invisible (NYI)
Pm = 30 / 40	fg/bg Black
Pm = 31 / 41	fg/bg Red
Pm = 32 / 42	fg/bg Green
Pm = 33 / 43	fg/bg Yellow
Pm = 34 / 44	fg/bg Blue
Pm = 35 / 45	fg/bg Magenta
Pm = 36 / 46	fg/bg Cyan
Pm = 37 / 47	fg/bg White
Pm = 38;5 / 48;5	set fg/bg to colour #m (ISO 8613-6)
Pm = 38;2;R;G;B	set fg to 24-bit colour #RGB (ISO 8613-3)
Pm = 48;2;R;G;B	set bg to 24-bit colour #RGB (ISO 8613-3)
Pm = 39 / 49	fg/bg Default
Pm = 90 / 100	fg/bg Bright Black
Pm = 91 / 101	fg/bg Bright Red
Pm = 92 / 102	fg/bg Bright Green
Pm = 93 / 103	fg/bg Bright Yellow
Pm = 94 / 104	fg/bg Bright Blue
Pm = 95 / 105	fg/bg Bright Magenta
Pm = 96 / 106	fg/bg Bright Cyan
Pm = 97 / 107	fg/bg Bright White
Pm = 99 / 109	fg/bg Bright Default

ESC [Ps n

Device Status Report (DSR)

Ps = 5	Status Report ESC [0 n (``OK")
Ps = 6	Report Cursor Position (CPR) [row;column] as ESC [r ; c R

Ps = 7	Request Display Name (insecure mode)
Ps = 8	Request Version Number (place in window title)

ESC [Ps SP q

Set Cursor Style (DECSCUSR)

Ps = 0	Blink Block
Ps = 1	Blink Block
Ps = 2	Steady Block
Ps = 3	Blink Underline
Ps = 4	Steady Underline
Ps = 5	Blink Bar (XTerm)
Ps = 6	Steady Bar (XTerm)

ESC [Ps;Ps r

Set Scrolling Region [top;bottom] [default: full size of window] (CSR)

ESC [s

Save Cursor (SC)

ESC [Ps;Pt t

Window Operations

Ps = 1	Deiconify (map) window
Ps = 2	Iconify window
Ps = 3	ESC [3 ; X ; Y t Move window to (X|Y)
Ps = 4	ESC [4 ; H ; W t Resize to WxH pixels
Ps = 5	Raise window
Ps = 6	Lower window
Ps = 7	Refresh screen once
Ps = 8	ESC [8 ; R ; C t Resize to R rows and C columns
Ps = 11	Report window state (responds with Ps = 1 or Ps = 2)
Ps = 13	Report window position (responds with Ps = 3)
Ps = 14	Report window pixel size (responds with Ps = 4)
Ps = 18	Report window text size (responds with Ps = 7)
Ps = 19	Currently the same as Ps = 18, but responds with Ps = 9
Ps = 20	Reports icon label (ESC] L NAME \ 234) (insecure mode)
Ps = 21	Reports window title (ESC] l NAME \ 234) (insecure mode)
Ps = 24..	Set window height to Ps rows

ESC [u

Restore Cursor

ESC [Ps x

Request Terminal Parameters (DECREQTPARM)

DEC Private Modes

ESC [? Pm h

DEC Private Mode Set (DECSET)

ESC [? Pm l

DEC Private Mode Reset (DECRST)

ESC [? Pm $ p

DEC Private Mode Request (DECRQM)

ESC [? Pm r

Restore previously saved DEC Private Mode Values.

ESC [? Pm s

Save DEC Private Mode Values.

ESC [? Pm t

Toggle DEC Private Mode Values (rxvt extension).

where

Pm = 1 (DECCKM)

h	Application Cursor Keys
l	Normal Cursor Keys

Pm = 2 (DECANM)

h	Enter VT52 mode
l	Enter VT52 mode

Pm = 3 (DECCOLM)

h	132 Column Mode
l	80 Column Mode

Pm = 4 (DECSCLM)

h	Smooth (Slow) Scroll
l	Jump (Fast) Scroll

Pm = 5 (DECSCNM)

h	Reverse Video
l	Normal Video

Pm = 6 (DECOM)

h	Origin Mode
l	Normal Cursor Mode

Pm = 7 (DECAWM)

h	Wraparound Mode
l	No Wraparound Mode

Pm = 8 (DECARM) *unimplemented*

h	Auto-repeat Keys
l	No Auto-repeat Keys

Pm = 9 (X10 XTerm mouse protocol)

h	Send Mouse X & Y on button press.
l	No mouse reporting.

Pm = 12 (AT&T 610, XTerm)

h	Blinking cursor (cvvis)
l	Steady cursor (cnorm)

Pm = 25 (DECTCEM)

h	Visible cursor {cnorm/cvvis}
l	Invisible cursor {civis}

Pm = 30 (rxvt)

h	scrollBar visible
l	scrollBar invisible

Pm = 35 (rxvt)

h	Allow XTerm Shift+key sequences
l	Disallow XTerm Shift+key sequences

Pm = 38 *unimplemented*

Enter Tektronix Mode (DECTEK)

Pm = 40

h	Allow 80/132 Mode
l	Disallow 80/132 Mode

Pm = 44 *unimplemented*

h	Turn On Margin Bell
l	Turn Off Margin Bell

Pm = 45 *unimplemented*

h	Reverse-wraparound Mode
l	No Reverse-wraparound Mode

Pm = 46 *unimplemented*

Pm = 47

h	Use Alternate Screen Buffer
l	Use Normal Screen Buffer

Pm = 66 (DECNKM)

h	Application Keypad (DECKPAM/DECPAM) == ESC =
l	Normal Keypad (DECKPNM/DECPNM) == ESC >

Pm = 67 (DECBKM)

h	Backspace key sends BS
l	Backspace key sends DEL

Pm = 1000 (X11 XTerm mouse protocol)

h	Send Mouse X & Y on button press and release.
l	No mouse reporting.

Pm = 1001 (X11 XTerm) *unimplemented*

h	Use Hilite Mouse Tracking.
l	No mouse reporting.

Pm = 1002 (X11 XTerm cell motion mouse tracking)

h	Send Mouse X & Y on button press and release, and motion with a button pressed.
l	No mouse reporting.

Pm = 1003 (X11 XTerm all motion mouse tracking)

h	Send Mouse X & Y on button press and release, and motion.
l	No mouse reporting.

Pm = 1004 (X11 XTerm focus in/focus out events)

h	Send Mouse focus in/focus out events.
l	Don't send focus events.

Pm = 1005 (X11 XTerm UTF-8 mouse mode) (Compile frills)

Try to avoid this mode, it doesn't work sensibly in non-UTF-8 locales. Use mode `1015` instead.

Unlike XTerm, coordinates larger than 2015 will work fine.

h	Enable mouse coordinates in locale-specific encoding.
l	Disable mouse coordinates in locale-specific encoding.

Pm = 1006 (X11 XTerm SGR mouse mode) (Compile frills)

h	Enable xterm SGR mouse coordinate reporting.
l	Disable xterm SGR mouse coordinate reporting.

Pm = 1010 (rxvt)

h	Don't scroll to bottom on TTY output
l	Scroll to bottom on TTY output

Pm = 1011 (rxvt)

h	Scroll to bottom when a key is pressed
l	Don't scroll to bottom when a key is pressed

Pm = 1015 (rxvt-unicode) (Compile frills)

h	Enable urxvt mouse coordinate reporting.
l	Disable urxvt mouse coordinate reporting.

Pm = 1021 (rxvt)

h	Bold/italic implies high intensity (see option -is)
l	Font styles have no effect on intensity (Compile styles)

Pm = 1047 (X11 XTerm alternate screen buffer)

h	Use Alternate Screen Buffer
l	Use Normal Screen Buffer - clear Alternate Screen Buffer if returning from it

Pm = 1048 (X11 XTerm alternate DECSC)

h	Save cursor position
l	Restore cursor position

Pm = 1049 (X11 XTerm 1047 + 1048)

h	Use Alternate Screen Buffer - clear Alternate Screen Buffer if switching to it
l	Use Normal Screen Buffer

Pm = 2004 (X11 XTerm bracketed paste mode)

h	Enable bracketed paste mode - prepend / append to the pasted text the control sequences ESC [200 ~ / ESC [201 ~
l	Disable bracketed paste mode

XTerm Operating System Commands

ESC] Ps;Pt ST

Set XTerm Parameters. 8-bit ST: 0x9c, 7-bit ST sequence: ESC \ (0x1b, 0x5c), backwards compatible terminator BEL (0x07) is also accepted. any **octet** can be escaped by prefixing it with SYN (0x16, ^V).

Many of these settings can be queried by specifying ? as parameter, but this requires insecure mode to be enabled for most of these.

Ps = 0	Change Icon Name and Window Title to Pt
Ps = 1	Change Icon Name to Pt
Ps = 2	Change Window Title to Pt
Ps = 3	If Pt starts with a ?, query the (STRING) property of the window and return it (insecure mode). If Pt contains a =, set the named property to the given value, else delete the specified property.
Ps = 4	Pt is a semi-colon separated sequence of one or more semi-colon separated number/name pairs, where number is an index to a colour and name is the name of a colour. Each pair causes the numbered colour to be changed to name. Numbers 0-7 corresponds to low-intensity (normal) colours and 8-15 corresponds to high-intensity colours. 0=black, 1=red, 2=green, 3=yellow, 4=blue, 5=magenta, 6=cyan, 7=white
Ps = 10	Change colour of text foreground to Pt
Ps = 11	Change colour of text background to Pt
Ps = 12	Change colour of text cursor foreground to Pt
Ps = 13	Change colour of mouse foreground to Pt
Ps = 17	Change background colour of highlight characters to Pt
Ps = 19	Change foreground colour of highlight characters to Pt
Ps = 20	Change background image to Pt (see the urxvt-background extension documentation)
Ps = 39	Change default foreground colour to Pt. [deprecated, use 10]
Ps = 46	Change Log File to Pt unimplemented
Ps = 49	Change default background colour to Pt. [deprecated, use 11]
Ps = 50	Set fontset to Pt, with the following special values of Pt (rxvt) #+n change up n #-n change down n if n is missing of 0, a value of 1 is used empty change to font0 n change to font n
Ps = 55	Log all scrollback buffer and all of screen to Pt [disabled]
Ps = 701	Change current locale to Pt, or, if Pt is ?, return the current locale (insecure mode, Compile frills).

Ps = 702	Request version if Pt is ?, returning rxvt-unicode, the resource name, the major and minor version numbers, e.g. ESC] 702 ; rxvt-unicode ; urxvt ; 7 ; 4 ST.
Ps = 704	Change colour of italic characters to Pt
Ps = 705	Change background tint color to Pt (see the urxvt-background extension documentation)
Ps = 706	Change colour of bold characters to Pt
Ps = 707	Change colour of underlined characters to Pt
Ps = 708	Change colour of the border to Pt
Ps = 710	Set normal fontset to Pt. Same as Ps = 50.
Ps = 711	Set bold fontset to Pt. Similar to Ps = 50 (Compile styles).
Ps = 712	Set italic fontset to Pt. Similar to Ps = 50 (Compile styles).
Ps = 713	Set bold-italic fontset to Pt. Similar to Ps = 50 (Compile styles).
Ps = 720	Move viewing window up by Pt lines, or clear scrollback buffer if Pt = 0 (Compile frills).
Ps = 721	Move viewing window down by Pt lines, or clear scrollback buffer if Pt = 0 (Compile frills).
Ps = 776	(urxvt 9.29) Returns info about the character cell size, replies with ESC] 776 ; cell-width ; cell-height ; font-ascent ST
Ps = 777	Call the perl extension with the given string, which should be of the form extension;parameters (Compile perl).

Mouse Reporting

When mouse reporting is enabled and none of the extended mouse modes (1005, 1006, 1015) is active, urxvt sends the following sequence on a mouse event:

```
ESC [ M <b> <x> <y>
```

The lower 2 bits of `<b>` indicate the button:

Button = `(<b> - SPACE) & 3`

0	Button1 pressed
1	Button2 pressed
2	Button3 pressed

3	button released (X11 mouse report)

The upper bits of **<b>** indicate the modifiers when the button was pressed and are added together (X11 mouse report only):

State = **(<b> - SPACE) & ~3**

4	Shift
8	Meta
16	Control
32	Motion Notify
32	Double Click (rxvt extension), disabled by default
64	Button1 is actually Button4, Button2 is actually Button5 etc.

x and y encode the coordinates (1 | 1 is the upper left corner, just as with cursor positioning):

Col = **<x> - SPACE**
Row = **<y> - SPACE**

The parameters include an offset of 32 to ensure that they are printable characters.

Example: Shift-Button-1 press at top row, column 80.

```
ESC [ M $ p !
```

The largest coordinate that can be represented in this encoding is 223. The range can be extended by using one of the extended mouse modes, which should be enabled *before* enabling mouse reporting, for semi-obvious reasons.

Mode 1005

If mode 1005 is active, urxvt sends the sequence

ESC [M <b> <x> <y>

with the coordinates provided as characters in locale-encoding instead of 1 byte octets. This mode does not work sensibly in non-UTF-8 locales and should therefore be avoided.

Mode 1006

If mode 1006 is active, urxvt sends the following sequences:

ESC [< <b>;<x>;<y> M

button press and motion

ESC [< <b>;<x>;<y> m

button release

where the parameters are provided as decimal numbers instead of octets and do not include an offset of 32.

The lower 2 bits of b encode the button number also on button release (instead of the value 3). The final character of the sequence (M or m) specifies the event type (press/motion or release).

Example: Shift-Button-1 press at top row, column 80.

```
ESC [ < 4 ; 80 ; 1 M
```

Mode 1015

If mode 1015 is active, urxvt sends the sequence

```
ESC [ <b>;<x>;<y> M
```

where the parameters are provided as decimal numbers instead of octets and only `b` includes an offset of 32.

Example: Shift-Button-1 press at top row, column 80.

```
ESC [ 36 ; 80 ; 1 M
```

Key Codes

Note: **Shift** + **F1**-**F10** generates **F11**-**F20**

For the keypad, use **Shift** to temporarily toggle Application Keypad mode and use **Num_Lock** to override Application Keypad mode, i.e. if **Num_Lock** is on the keypad is in normal mode. Also note that the values of **BackSpace**, **Delete** may have been compiled differently on your system.

	Normal	Shift	Control	Ctrl+Shift
Tab	^I	ESC [Z	^I	ESC [Z
BackSpace	^?	^?	^H	^H
Find	ESC [1 ~	ESC [1 $	ESC [1 ^	ESC [1 @
Insert	ESC [2 ~	paste	ESC [2 ^	ESC [2 @
Execute	ESC [3 ~	ESC [3 $	ESC [3 ^	ESC [3 @
Select	ESC [4 ~	ESC [4 $	ESC [4 ^	ESC [4 @
Prior	ESC [5 ~	scroll-up	ESC [5 ^	ESC [5 @
Next	ESC [6 ~	scroll-down	ESC [6 ^	ESC [6 @
Home	ESC [7 ~	ESC [7 $	ESC [7 ^	ESC [7 @
End	ESC [8 ~	ESC [8 $	ESC [8 ^	ESC [8 @
Delete	ESC [3 ~	ESC [3 $	ESC [3 ^	ESC [3 @
F1	ESC [11 ~	ESC [23 ~	ESC [11 ^	ESC [23 ^
F2	ESC [12 ~	ESC [24 ~	ESC [12 ^	ESC [24 ^
F3	ESC [13 ~	ESC [25 ~	ESC [13 ^	ESC [25 ^
F4	ESC [14 ~	ESC [26 ~	ESC [14 ^	ESC [26 ^
F5	ESC [15 ~	ESC [28 ~	ESC [15 ^	ESC [28 ^
F6	ESC [17 ~	ESC [29 ~	ESC [17 ^	ESC [29 ^
F7	ESC [18 ~	ESC [31 ~	ESC [18 ^	ESC [31 ^
F8	ESC [19 ~	ESC [32 ~	ESC [19 ^	ESC [32 ^
F9	ESC [20 ~	ESC [33 ~	ESC [20 ^	ESC [33 ^
F10	ESC [21 ~	ESC [34 ~	ESC [21 ^	ESC [34 ^
F11	ESC [23 ~	ESC [23 $	ESC [23 ^	ESC [23 @
F12	ESC [24 ~	ESC [24 $	ESC [24 ^	ESC [24 @
F13	ESC [25 ~	ESC [25 $	ESC [25 ^	ESC [25 @
F14	ESC [26 ~	ESC [26 $	ESC [26 ^	ESC [26 @
F15 (Help)	ESC [28 ~	ESC [28 $	ESC [28 ^	ESC [28 @
F16 (Menu)	ESC [29 ~	ESC [29 $	ESC [29 ^	ESC [29 @
F17	ESC [31 ~	ESC [31 $	ESC [31 ^	ESC [31 @
F18	ESC [32 ~	ESC [32 $	ESC [32 ^	ESC [32 @
F19	ESC [33 ~	ESC [33 $	ESC [33 ^	ESC [33 @
F20	ESC [34 ~	ESC [34 $	ESC [34 ^	ESC [34 @
				Application
Up	ESC [A	ESC [a	ESC O a	ESC O A

Down	ESC [B	ESC [b	ESC O b	ESC O B
Right	ESC [C	ESC [c	ESC O c	ESC O C
Left	ESC [D	ESC [d	ESC O d	ESC O D
KP_Enter	^M			ESC O M
KP_F1	ESC O P			ESC O P
KP_F2	ESC O Q			ESC O Q
KP_F3	ESC O R			ESC O R
KP_F4	ESC O S			ESC O S
KP_Multiply	*			ESC O j
KP_Add	+			ESC O k
KP_Separator	,			ESC O l
KP_Subtract	-			ESC O m
KP_Decimal	.			ESC O n
KP_Divide	/			ESC O o
KP_0	0			ESC O p
KP_1	1			ESC O q
KP_2	2			ESC O r
KP_3	3			ESC O s
KP_4	4			ESC O t
KP_5	5			ESC O u
KP_6	6			ESC O v
KP_7	7			ESC O w
KP_8	8			ESC O x
KP_9	9			ESC O y

CONFIGURE OPTIONS

General hint: if you get compile errors, then likely your configuration hasn't been tested well. Either try with `--enable-everything` or use the default configuration (i.e. no `--enable-xxx` or `--disable-xxx` switches). Of course, you should always report when a combination doesn't work, so it can be fixed. Marc Lehmann <rxvt@schmorp.de>.

All

--enable-everything

Add (or remove) support for all non-multichoice options listed in `./configure --help`, except for `--enable-assert` and `--enable-256-color`.

You can specify this and then disable options you do not like by *following* this with the appropriate `--disable-...` arguments, or you can start with a minimal configuration by specifying `--disable-everything` and than adding just the `--enable-...` arguments you want.

--enable-xft (default: on)

Add support for Xft (anti-aliased, among others) fonts. Xft fonts are slower and require lots of memory, but as long as you don't use them, you don't pay for them.

--enable-font-styles (default: on)

Add support for **bold**, *italic* and ***bold italic*** font styles. The fonts can be set manually or automatically.

--with-codesets=CS,... (default: all)

Compile in support for additional codeset (encoding) groups (`eu`, `vn` are always compiled in, which includes most 8-bit character sets). These codeset tables are used for driving X11 core fonts, they are not required for Xft fonts, although having them compiled in lets rxvt-unicode choose replacement fonts more intelligently. Compiling them in will make your

binary bigger (all of together cost about 700kB), but it doesn't increase memory usage unless you use a font requiring one of these encodings.

all	all available codeset groups
zh	common chinese encodings
zh_ext	rarely used but very big chinese encodings
jp	common japanese encodings
jp_ext	rarely used but big japanese encodings
kr	korean encodings

–enable-xim (default: on)

Add support for XIM (X Input Method) protocol. This allows using alternative input methods (e.g. kinput2) and will also correctly set up the input for people using dead keys or compose keys.

–enable-unicode3 (default: off)

Recommended to stay off unless you really need a lot of non-BMP characters.

Enable support for direct storage of unicode characters above 65535 (the basic multilingual page). This increases storage requirements per character from 2 to 4 bytes. X11 fonts do not yet support these extra characters, but Xft does.

Please note that rxvt-unicode can store and display unicode characters above 65535 even without this flag, but the number of such characters is limited to a few thousand (shared with combining characters, see next switch).

–enable-combining (default: on)

Enable automatic composition of combining characters into composite characters. This is required for proper viewing of text where accents are encoded as separate unicode characters. This is done by using precomposed characters when available or creating new pseudo-characters when no precomposed form exists.

Without –enable-unicode3, the number of additional precomposed characters is somewhat limited (the 6400 private use characters will be (ab-)used). With –enable-unicode3, no practical limit exists.

This option will also enable storage (but not display) of characters beyond plane 0 (>65535) when –enable-unicode3 was not specified.

The combining table also contains entries for arabic presentation forms, but these are not currently used. Bug me if you want these to be used (and tell me how these are to be used...).

–enable-fallback[=CLASS] (default: Rxvt)

When reading resource settings, also read settings for class CLASS. To disable resource fallback use –disable-fallback.

–with-res-name=NAME (default: urxvt)

Use the given name as default application name when reading resources. Specify –with-res-name=rxvt to replace rxvt.

–with-res-class=CLASS (default: URxvt)

Use the given class as default application class when reading resources. Specify –with-res-class=Rxvt to replace rxvt.

–enable-pixbuf (default: on)

Add support for GDK-PixBuf to be used for background images. It adds support for many file formats including JPG, PNG, TIFF, GIF, XPM, BMP, ICO and TGA.

–enable-startup-notification (default: on)

Add support for freedesktop startup notifications. This allows window managers to display some kind of progress indicator during startup.

–enable-transparency (default: on)

Add support for using the root pixmap as background to simulate transparency. Note that this feature depends on libXrender and on the availability of the RENDER extension in the X server.

–enable-fading (default: on)

Add support for fading the text when focus is lost.

–enable-rxvt-scroll (default: on)

Add support for the original rxvt scrollbar.

–enable-next-scroll (default: on)

Add support for a NeXT-like scrollbar.

–enable-xterm-scroll (default: on)

Add support for an Xterm-like scrollbar.

–disable-backspace-key

Removes any handling of the backspace key by us - let the X server do it.

–disable-delete-key

Removes any handling of the delete key by us - let the X server do it.

–disable-resources

Removes any support for resource checking.

–disable-swapscreen

Remove support for secondary/swap screen.

–enable-frills (default: on)

Add support for many small features that are not essential but nice to have. Normally you want this, but for very small binaries you may want to disable this.

A non-exhaustive list of features enabled by `--enable-frills` (possibly in combination with other switches) is:

```
MWM-hints
EWMH-hints (pid, utf8 names) and protocols (ping)
urgency hint
separate underline colour (-underlineColor)
settable border widths and borderless switch (-w, -b, -bl)
visual depth selection (-depth)
settable extra linespacing (-lsp)
iso-14755 5.1 (basic) support
tripleclickwords (-tcw)
settable insecure mode (-insecure)
keysym remapping support
cursor blinking and underline cursor (-bc, -uc)
XEmbed support (-embed)
user-pty (-pty-fd)
hold on exit (-hold)
compile in built-in block graphics
skip builtin block graphics (-sbg)
separate highlight colour (-highlightColor, -highlightTextColor)
focus reporting mode (1004).
extended mouse reporting modes (1005, 1006 and 1015).
visual selection via -visual and -depth.
systemd socket activation
selectable rewrapmode
bracketed paste mode
```

It also enables some non-essential features otherwise disabled, such as:

```
some round-trip time optimisations
nearest colour allocation on pseudocolor screens
UTF8_STRING support for selection
sgr modes 90..97 and 100..107
backindex and forwardindex escape sequences
view change/zero scrollback escape sequences
locale switching escape sequence
window op and some xterm/OSC escape sequences
rectangular selections
trailing space removal for selections
verbose X error handling
```

–enable-iso14755 (default: on)

Enable extended ISO 14755 support (see urxvt(1)). Basic support (section 5.1) is enabled by `--enable-frills`, while support for 5.2, 5.3 and 5.4 is enabled with this switch.

–enable-keepscrolling (default: on)

Add support for continual scrolling of the display when you hold the mouse button down on a scrollbar arrow.

–enable-selectionscrolling (default: on)

Add support for scrolling when the selection moves to the top or bottom of the screen.

–enable-mousewheel (default: on)

Add support for scrolling via mouse wheel or buttons 4 & 5.

–enable-slipwheeling (default: on)

Add support for continual scrolling (using the mouse wheel as an accelerator) while the control key is held down. This option requires –enable-mousewheel to also be specified.

–enable-smart-resize (default: off)

Add smart growth/shrink behaviour when resizing. This should keep the window corner which is closest to a corner of the screen in a fixed position.

–enable-text-blink (default: on)

Add support for blinking text.

–enable-pointer-blank (default: on)

Add support to have the pointer disappear when typing or inactive.

–enable-perl (default: on)

Enable an embedded perl interpreter. See the **urxvtperl(3)** manpage for more info on this feature, or the files in *src/perl/* for the extensions that are installed by default. The perl interpreter that is used can be specified via the `PERL` environment variable when running configure. Even when compiled in, perl will *not* be initialised when all extensions have been disabled `-pe "" --perl-ext-common ""`, so it should be safe to enable from a resource standpoint.

–enable-assert (default: off)

Enables the assertions in the code, normally disabled. This switch is only useful when developing rxvt-unicode.

–enable-256-color (default: off)

Force use of so-called 256 colour mode, to work around buggy applications that do not support termcap/terminfo, or simply improve support for applications hardcoding the xterm 256 colour table.

This switch breaks termcap/terminfo compatibility to `TERM=rxvt-unicode`, and consequently sets `TERM` to `rxvt-unicode-256color` by default (*doc/etc/* contains termcap/terminfo definitions for both).

It also results in higher memory usage and can slow down urxvt dramatically when more than six fonts are in use by a terminal instance.

–with-name=NAME (default: urxvt)

Set the basename for the installed binaries, resulting in `urxvt`, `urxvtd` etc.). Specify `--with-name=rxvt` to replace with `rxvt`.

–with-term=NAME (default: rxvt-unicode)

Sets the default `TERM` value that urxvt sets. The default is either `rxvt-unicode` or `rxvt-unicode-256color`, as appropriate.

–with-terminfo=PATH

If set, urxvt will set the environment variable `TERMINFO` to the given PATH, which can be useful as a last resort if installing the terminfo entries system-wide is not possible.

–with-x

Use the X Window System (pretty much default, eh?).

AUTHORS

Marc Lehmann <rxvt@schmorp.de> converted this document to pod and reworked it from the original Rxvt documentation, which was done by Geoff Wing <gcw@pobox.com>, who in turn used the XTerm documentation and other sources.