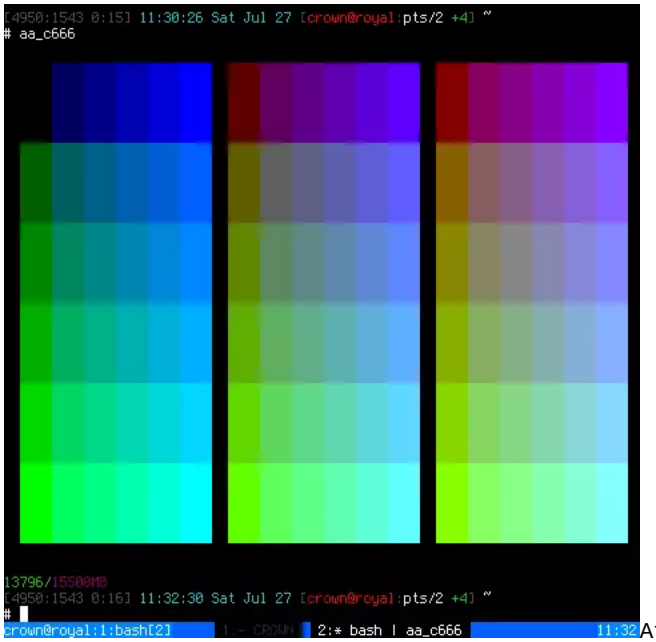
- Skip to main content
- Skip to search

Home → Linux → RXVT Customization with ~/.Xresources

RXVT Customization with ~/.Xresources



After many years of using all sorts of terminal emulators, from xterm to the Gnome terminal, to KDE Konsole to xfce4-terminal, **lxterminal**, vte, yakuake, rote, roxterm, putty, sakura, **terminator**, and finally I settled in for the long-haul with rxvt (rxvt-unicode) aka **urxvt**.

I have X setup to start urxvt on startup, and have urxvt setup to auto-start a tmux session. It's freaking sweet.

The customizations I have made to my urxvt are through the use of X resources.

My BOX: **Slim -> Ratpoison -> URxvt -> Tmux -> Bash**

Download my ~/.Xresources file.

Table of Contents [hide]

Table of Contents

rxvt-unicode (urxvt)

rxvt-unicode is a highly customizable terminal emulator forked from rxvt. Commonly known as urxvt, rxvt-unicode can be daemonized to run clients within a single process in order to minimize the use of system resources. Developed by Marc Lehmann, some of the more outstanding features of rxvt-unicode include international language support through Unicode, the ability to display multiple font types and support for Perl extensions. Also see: rxvt-unicode wiki page on ArchWiki.

Xresources vs. Xdefaults

> Definitive Verbose Answer
>
> `~/.Xdefaults` is the older method of storing X resources. This file is re-read every time an Xlib program is started. If X11 is used over the network, the file must be present on the same filesystem as the programs. `~/.Xresources` is newer. It is loaded with xrdb into the `RESOURCE_MANAGER` property of the X11 root window. Whenever any program looks up a resource, it is read straight from `RESOURCE_MANAGER`. If this property does not exist, Xlib falls back to the old method of reading `.Xdefaults` on every program startup. Note that most distributions will load `~/.Xresources` automatically if it is present, causing `.Xdefaults` to be ignored even if you have never run xrdb manually. The advantage of the new method is that it's enough to call xrdb once, and the resources will be available to any program running on this display, whether local or remote. (The name `~/.Xresources` is only a convention – you can use xrdb to load any file, even `.Xdefaults`.)

Adding to .Xresources

You can stick this in your `~/.xinitrc` file if need be.

```
[[ -f ~/.Xdefaults ]] && xrdb -merge ~/.Xdefaults
```

After you make a change to your ~/.Xdefaults or ~/.Xresources file you will need to reload it with: `xrdb ~/.Xdefaults` and then close rxvt and reopen.

Useful Xresources for rxvt

Note that I use the 256-color enabled unicode version, named rxvt-unicode, so the name is **URxvt**, you could also try **Rxvt** as the resource name prefix.

Download my ~/.Xresources file.

```
URxvt*termName: screen-256color
URxvt*geometry: 240x84
URxvt*loginShell: true
URxvt*scrollColor: #777777
URxvt*scrollstyle: plain
URxvt*scrollTtyKeypress: true
URxvt*scrollTtyOutput: false
URxvt*scrollWithBuffer: false
URxvt*secondaryScreen: true
URxvt*secondaryScroll: true
URxvt*skipScroll: true
URxvt*scrollBar: false
URxvt*scrollBar_right: false
URxvt*scrollBar_floating: false
URxvt*fading: 30
URxvt*utmpInhibit: false
URxvt*urgentOnBell: false
URxvt*visualBell: true
URxvt*mapAlert: true
URxvt*mouseWheelScrollPage: false
URxvt*background: Black
URxvt*foreground: White
URxvt*colorUL: yellow
URxvt*underlineColor: yellow
URxvt*font: -xos4-terminus-medium-*-*-*-14-*-*-*-*-*-iso8859-15,xft:terminus:pixelsize:12
URxvt*boldFont: -xos4-terminus-bold-*-*-*-14-*-*-*-*-*-iso8859-15,xft:terminus:bold:pixelsize:12
URxvt*italicFont: xft:Bitstream Vera Sans Mono:italic:autohint=true:pixelsize=12
URxvt*boldItalicFont: xft:Bitstream Vera Sans Mono:bold:italic:autohint=true:pixelsize=12
URxvt*saveLines: 0
URxvt*buffered: true
URxvt*hold: false
URxvt*internalBorder:
URxvt*print-pipe: cat > $HOME/$(echo urxvt.dump.$(date +'%Y%M%d%H%m%S'))
```

```
URxvt*perl-ext-common:
URxvt*perl-ext:
```

termName

Specifies the terminal type name to be set in the TERM environment variable. I use tmux so this is helpful.

```
URxvt*termName:  screen-256color
```

geometry

Create the window with the specified X window geometry [default 80x24].. base it on your `$LINES` and `$COLUMNS`.

```
URxvt*geometry:  240x84
```

loginShell

Start as a login shell by prepending a - to argv[0] of the shell. Again, for tmux, this is super-helpful and causes your bash login files like `~/.bash_profile` to be loaded.

```
URxvt*loginShell:  true
```

scrollTtyKeypress

True: scroll to bottom when a non-special key pressed. Special keys are those which are intercepted by rxvt for special handling andnt passed onto the shell

```
URxvt*scrollTtyKeypress:  true
```

scrollTtyOutput

Do not scroll to bottom when tty receives output

```
URxvt*scrollTtyOutput:  false
```

scrollWithBuffer

Do not scroll with scrollback buffer when tty recieves new lines, adds some speed.. also, I use tmux scrollback buffers.

```
URxvt*scrollWithBuffer:  false
```

skipScroll

For speed. When receiving lots of lines, urxvt will only scroll once in a while (around 60x/sec), resulting in fewer updates. This can result in urxvt not ever displaying some of the lines it receives

```
URxvt*skipScroll:  true
```

scrollBar

Disable the scrollbar.. why waste valuable screen real-estate when you should be using tmux scrollback?

```
URxvt*scrollBar:  false
```

fading

Fade the text by the given percentage when focus is lost. This is neat, when I switch to a different window, or switch to a different machine ala synergy, it will fade the screen slightly.

```
URxvt*fading:  30
```

visualBell

Use visual bell on receipt of a bell character. Helpful to be used with inputrc and tmux.

```
URxvt*visualBell:  true
```

background

Use the specified colour as the windows background colour [default White]. Why in the world would you default white unless you are old-school... as in 70s.

```
URxvt*background:  Black
```

foreground

Use the specified colour as the windows foreground colour [default Black]. see above.

```
URxvt*foreground:  White
```

colorUL

Use the specified colour to disp1ay underlined characters when the foreground colour is the default. Makes it easier to notice, rxvt-unicode authors choice as well.

```
URxvt*colorUL:  yellow
```

underlineColor

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

```
URxvt*underlineColor:  yellow
```

font, boldFont, italicFont, boldItalicFont

A comma separated list of font names that are checked in order when trying to find glyphs for characters. Man for coding, nothing beats the terminus font.. nothing! Also, notice that boldFont, italicFont, and boldItalicFont are specified as well. This makes a huge difference you will notice right away.

```
URxvt*font:  -xos4-terminus-medium-*-*-*-14-*-*-*-*-*-iso8859-15,xft:terminus:pixelsize:12
URxvt*boldFont:  -xos4-terminus-bold-*-*-*-14-*-*-*-*-*-iso8859-15,xft:terminus:bold:pixelsize:12
URxvt*italicFont:  xft:Bitstream Vera Sans Mono:italic:autohint=true:pixelsize=12
URxvt*boldItalicFont:  xft:Bitstream Vera Sans Mono:bold:italic:autohint=true:pixelsize=12
```

saveLines

Save number lines in the scrollback buffer [default 64]. This resource is limited on most machines to 65535. I am a power-user, so I always use a multiplexer. Tmux if its available, otherwise screen. So I use the scrollback buffer in tmux or screen, which is much nicer.

```
URxvt*saveLines:  0
```

print-pipe

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

```
URxvt*print-pipe:  cat > $HOME/$(echo urxvt.dump.$(date +'%Y%M%d%H%m%S'))
```

perl-ext

Comma-separated list(s) of perl extension scripts (default: "default") to use in this terminal instance, blank disables. By setting these both to blank, it completely disables perl from being initialized, thus much faster and smaller footprint. Plus it is more secure.

```
URxvt*perl-ext:
URxvt*perl-ext-common:
```

Output RXVT Resources

This simple command is built-in to rxvt to show a list of Resource Names. Useful for pasting into your `~/.Xresources` or `~/.Xdefaults`

```
# urxvt --help 2>&1| sed -n '/:  /s/^ */! URxvt*/gp'


! URxvt*termName:                      string
! URxvt*geometry:                      geometry
! URxvt*chdir:                         string
! URxvt*reverseVideo:                  boolean
! URxvt*loginShell:                    boolean
! URxvt*jumpScroll:                    boolean
! URxvt*skipScroll:                    boolean
! URxvt*pastableTabs:                  boolean
! URxvt*scrollstyle:                   mode
! URxvt*scrollBar:                     boolean
! URxvt*scrollBar_right:               boolean
! URxvt*scrollBar_floating:            boolean
! URxvt*scrollBar_align:               mode
! URxvt*thickness:                     number
! URxvt*scrollTtyOutput:               boolean
! URxvt*scrollTtyKeypress:             boolean
! URxvt*scrollWithBuffer:              boolean
! URxvt*inheritPixmap:                 boolean
! URxvt*transparent:                   boolean
! URxvt*tintColor:                     color
```

```
! URxvt*shading:                      number
! URxvt*blurRadius:                   HxV
! URxvt*fading:                       number
! URxvt*fadeColor:                    color
! URxvt*utmpInhibit:                  boolean
! URxvt*urgentOnBell:                 boolean
! URxvt*visualBell:                   boolean
! URxvt*mapAlert:                     boolean
! URxvt*meta8:                        boolean
! URxvt*mouseWheelScrollPage:         boolean
! URxvt*tripleclickwords:             boolean
! URxvt*insecure:                     boolean
! URxvt*cursorUnderline:              boolean
! URxvt*cursorBlink:                  boolean
! URxvt*pointerBlank:                 boolean
! URxvt*background:                   color
! URxvt*foreground:                   color
! URxvt*color0:                       color
! URxvt*color1:                       color
! URxvt*color2:                       color
! URxvt*color3:                       color
! URxvt*color4:                       color
! URxvt*color5:                       color
```

rxvt Resources with descriptions

And here is a really helpful command to output the entire rxvt resources with descriptions taken from the manpage. I start by running this and appending it to the `~/.Xresources` file.

```
# man -Pcat urxvt | sed -n '/th: b/,/^B/p'|sed '$d'|sed '/^ \{7\}[a-z]/s/^ */^/g' | sed -e :a -e 'N;s/\n/@@/g;ta;P;D' | sed 's,\^\
([^@]\+\)@*[\t ]*\([^\^]\+\),! \2\n! URxvt*\1\n\n,g' | sed 's,@@\(  \+\),\n\1,g' | sed 's,@*$,,g' | sed '/^[^!]/d' | tr -d "'\`"

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

! 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.
! URxvt*buffered: boolean
```

Lean 256 Color Output Function

I've actually spent a lot of time on this function, it's gotta be the fastest leanest code on the net for displaying all 256 colors in a very useful way.

```
function aa_256 ()
{
    local o= i= x=`tput op` cols=`tput cols` y= oo= yy=;
    y=`printf %$(($cols-6))s`;
    yy=${y// /=};
    for i in {0..256};
    do
        o=00${i};
        oo=`echo -en "setaf ${i}\nsetab ${i}\n"|tput -S`;
        echo -e "${o:${#o}-3:3} ${oo}${yy}${x}";
    done
}
```

256 Colors with aa_c666

In case you wondered how I generated that screenshot.. here's a BASH function I wrote. Also see my extremely verbose post: [Terminal Escape Code Zen].

```
function aa_c666 ()
{
    local r= g= b= c= CR="`tput sgr0;tput init`" C="`tput op`" n="\n\n\n" t="  " s="    ";
    echo -e "${CR}${n}";
    function c666 ()
    {
        local b= g=$1 r=$2;
        for ((b=0; b<6; b++))
        do
            c=$(( 16 + ($r*36) + ($g*6) + $b ));
            echo -en "setaf ${c}\nsetab ${c}\n" | tput -S;
            echo -en "${s}";
        done
    };
    function c666b ()
    {
        local g=$1 r=;
        for ((r=0; r<6; r++))
        do
            echo -en " `c666 $g $r`${C} ";
        done
    };
    for ((g=0; g<6; g++))
    do
```

```
        c666b=`c666b $g`;
        echo -e " ${c666b}";
        echo -e " ${c666b}";
        echo -e " ${c666b}";
        echo -e " ${c666b}";
        echo -e " ${c666b}";
    done;
    echo -e "${CR}${n}${n}"
}
```