Jafrog's dev blog

23 November 2013

Colors In Terminal

If you use terminal on a daily basis I bet you played with some color settings at least once. Colorizing `ls` output, shell prompt, `git` logs - those are very common tasks. And it's not surprising as color helps us parse information faster, pay attention to important parts and generally makes things prettier. Everybody loves pretty things, especially the ones you have to look at every day.

This post is an attempt to gather in one place and structure things I know about colors in terminal.

Escape sequences

You've probably seen things like `\e[32m` or `\x1b[1;31m`. These are ANSI escape codes used for defining a color. All ANSI escape sequences start with, well, `ESC`. There're several ways of encoding an `ESC`:

Shell	`\e`
ASCII Hex	`\0x1B`
ASCII Oct	`\033`

So `\x1b[31;4m`, `\e[31;4m` and `\033[31;4m` are different ways to write the same sequence. Let's look at the structure of this sequence.

`\x1b[` is a **Control Sequence Introducer** that consists of hexadecimal ASCII `ESC` character code and a `[`.

`31;4` is a list of instructions separated by `;`. Usually this list is formatted as follows:

```
[<PREFIX>];[<COLOR>];[<TEXT DECORATION>]
```

For example `31;4` means "no prefix, color - red, underline". `<PREFIX>` is used for 256 color mode. More on color modes later.

Finally `m` indicates the end of control sequence so terminal would know not to interpret text after `m` as a color code.

The following command should print "hello" in red underscore text:

```
> echo "\x1b[31;4mHello\x1b[0m"
```

`\x1b[0m` means "reset all attributes".

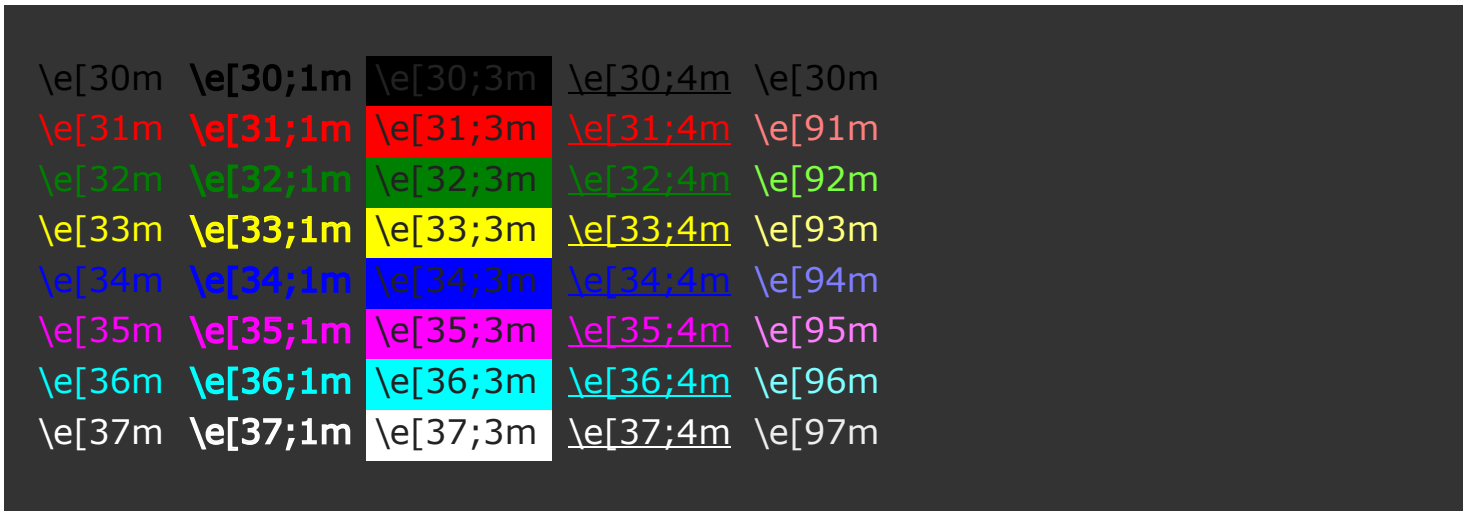
Color codes

Back in the old days terminals were different. Some of them could display only 16 colors and some of them went as far as 256. Now you probably work with a terminal emulator that runs on a machine that could display more than 16 million colors. But as terminal applications emulate older terminals, they usually support far less colors. For example a terminal app could be set up with 16 colors support or 256 colors support. The exact values of those colors depend on a terminal's settings.

To list all available colors in 16-color mode run:

```
> for code in {30..37}; do \
echo -en "\e[${code}m"'\\e['"$code"'m'"\e[0m"; \
echo -en "  \e[$code;1m"'\\e['"$code"';1m'"\e[0m"; \
echo -en "  \e[$code;3m"'\\e['"$code"';3m'"\e[0m"; \
echo -en "  \e[$code;4m"'\\e['"$code"';4m'"\e[0m"; \
echo -e "  \e[$((code+60))m"'\\e['"$((code+60))"'m'"\e[0m"; \
done
```

You should see something like this:



Or, in fact, something completely different! You can set up red to look like blue but I wouldn't recommend such a deception as it probably mess up some color themes in `bash`, `zsh` or anything else that runs in a terminal.

As you can see from command's output there're several sets of color codes:

Basic 8 colors	30..37
Basic "high contrast" colors	90..97
xterm-256 colors	0..255

And a set of text decoration indicators that should be placed right after the color code:

Bold	1
Underscore	4
Background	3

If color code is prefixed by `38;5` it is interpreted as one of 256 colors. E.g. `\e[38;5;91m` will color following text purple, while `\e[91m` will indicate bright red.

There're several 256 color palettes out there. For example, a couple of the popular ones are Web safe colors and X11 colors. And though they're both include 256 colors, they're two different palettes!

To enable 256 colors support, you have to set up your terminal as `xterm-256color` (in `iTerm2` go to **Preferences** > **Profiles** > **Terminal** > **Report Terminal Type** and put `xterm-256color` into the field). The set of colors you'll get is xterm-256 pallete. In some places, like Mac OSX default color picker, this palette is called "emacs", though it doesn't have anything to do with Emacs.

To list all colors available in 256 color mode with their codes run

```
> for code in {0..255}
    do echo -e "\e[38;5;${code}m"'\\e[38;5;'"$code"m"\e[0m"
  done
```

Bonus: To list all colors available in `Emacs` run `M-x list-colors-display`.

Thanks for the great and concise post. I have been wrestling with Vim/Tmux/Mintty trying to get everything properly Solarized, and I was stuck on getting the high 8 colors in 16-color mode. I don't know if I just skipped past in everything I've read, but I have not seen the bright/90 color code discussed anywhere else. I was using mintty's "bold as color" option to get my prompt how I wanted, but then I had problems using bold in vim. Now everything looks great, so thanks.

1 ▲ ▼ · Reply · Share ›

Victor Klimov · a year ago

Thanks, Irina! Very interesting!

1 ▲ ▼ · Reply · Share ›

Evgeny · 3 years ago

This is an awesomely clear guide on the terminal colors (and thanks to this clarity I understand how 'underline' works)! Thank you.
Btw, just out of curiosity — are the 3 types of escape sequences perfect substitutes or not? I'm using '\e' everywhere because it's the shortest and easiest to read, but don't understand whether there could be instances when it wouldn't work.

1 ▲ ▼ · Reply · Share ›

> **jafrog** Mod → Evgeny · 2 years ago
>
> It depends on what characters a particular terminal interprets as an escape sequence. AFAIK most modern terminal emulators recognise the \e sequence so you probably won't run into any trouble using it.
>
> ▲ ▼ · Reply · Share ›
>
> > **Evgeny** → jafrog · 2 years ago
> >
> > Yep, it mostly works, I've only run into trouble when was using '\e' within a function later used in my bash in default OSX terminal app PS1 to change color on error. Had to use '\033' to fix it (echo -ne "\033[1;32m").
> > Strangely enough, I use another function within PS1 that changes terminal title and it works just fine with '\e' (echo -ne "\e]2;${PWD/#$HOME/\~}\007")!
> >
> > And all color codes set directly in PS1 are working just fine with '\e' escape codes.
> > Strange, though not a big deal, since mistakes are plainly visible and easy to fix.
> >
> > ▲ ▼ · Reply · Share ›

Cristiano Vilela Moreira · 3 years ago

Excelent!!!

1 ▲ ▼ · Reply · Share ›

k-five · 3 years ago

very good
like +
:)

1 ▲ ▼ · Reply · Share ›

Jakub · a month ago

THIS IS GREAT!!!! I've been trying to find such an explanation for a very long time, and it was always not clear. You are genius!!!! Thanks.

▲ ▼ · Reply · Share ›

John Smith · 7 months ago

thnx! good write up

▲ ▼ · Reply · Share ›