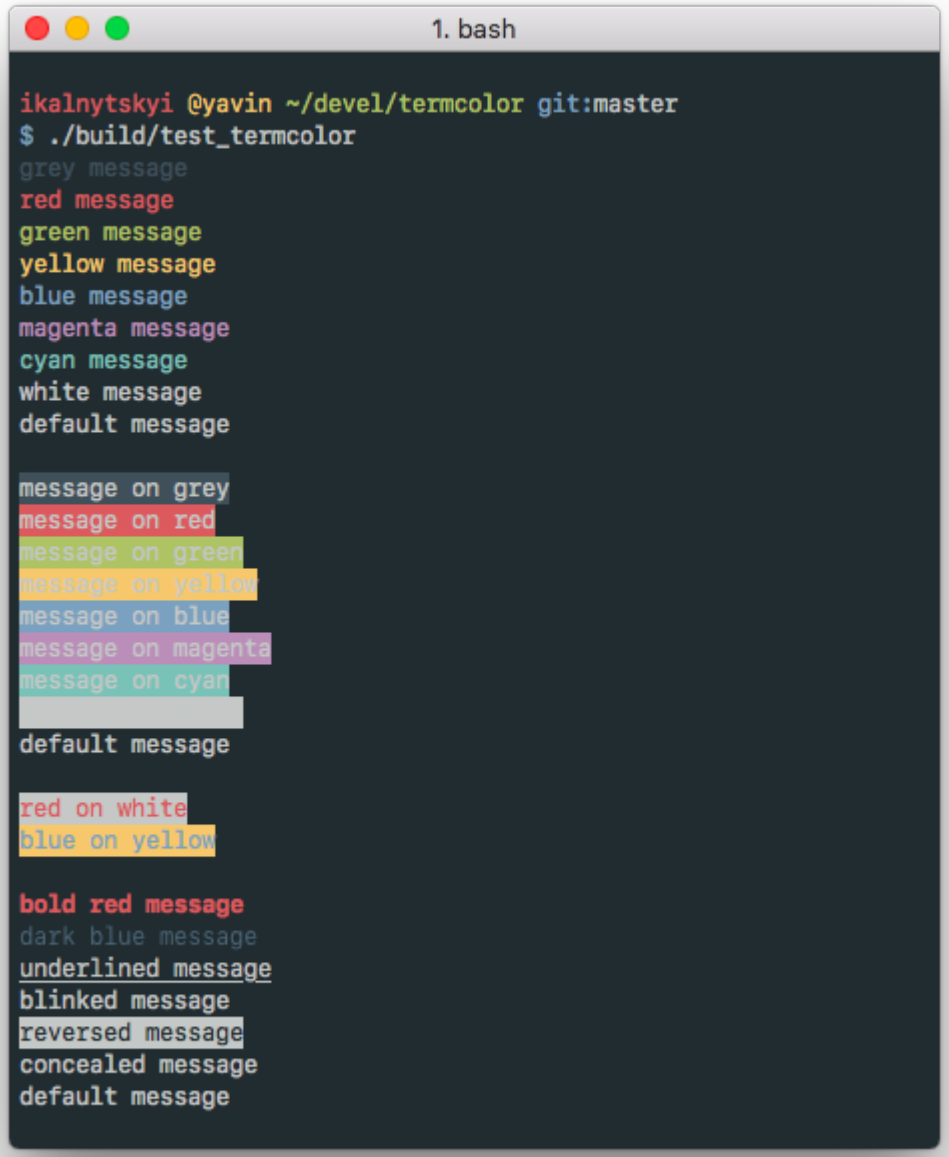
termcolor

Release 0.1

unknown

Jul 23, 2022

CONTENTS



Termcolor is a header-only C++ library for printing colored messages to the terminal. Written just for fun with a help of the Force. Termcolor uses ANSI color formatting, so you can use it on every system that is used such terminals (most *nix systems, including Linux and Mac OS).

Note: On Windows, Windows API is used instead of escape codes but some limitations are applied (not everything is supported). That's why it's recommended to enter virtual terminal processing mode and set `TERMCOLOR_USE_ANSI_ESCAPE_SEQUENCES` macro to trick termcolor to use ANSI color codes.

It's licensed under the BSD (3-clause) License. That basically means: do whatever you want as long as copyright sticks

around.

CHAPTER

ONE

INSTALLATION

- Add `termcolor.hpp` (grab it from `include/termcolor/termcolor.hpp`) to the project and use stream manipulators from the `termcolor` namespace.
- You can also use vcpkg to install the library:

```
$ vcpkg install termcolor
```

- Or if you are on macOS, you can use Homebrew for that purpose:

```
$ brew install termcolor
```

- For up-to-date information about existing packages, refer to the the following picture:

CHAPTER

TWO

HOW TO USE?

It's very easy to use. The idea is built upon C++ stream manipulators. Typical «Hello World» application looks like this:

```
#include <iostream>
#include <termcolor/termcolor.hpp>

int main(int /*argc*/, char** /*argv*/)
{
    std::cout << termcolor::red << "Hello, ";                   // 16 colors
    std::cout << termcolor::color<100> << "Colorful ";          // 256 colors
    std::cout << termcolor::color<211, 54, 130> << "World!";    // true colors
    std::cout << std::endl;
    return 0;
}
```

The application above prints a string using different colors. There is one caveat though. You must not forget to reset colors, otherwise they will be applied to other prints as well.

```
std::cout << termcolor::red << "Hello, Colorful World!" << std::endl;
std::cout << "I'm RED too!" << std::endl;
```

Correct version of the code above should look like this:

```
std::cout << termcolor::red << "Hello, Colorful World!" << termcolor::reset << std::endl;
std::cout << termcolor::reset << "Here I'm!" << std::endl;
```

By default, Termcolor ignores any colors for non-tty streams (e.g. `std::stringstream`), so the following snippet

```
std::stringstream ss;
ss << termcolor::red << "unicorn";
std::cout << ss.str();
```

will print «unicorn» using default color, not red. In order to change this behaviour one can use `termcolor::colorize` manipulator that enforce colors no matter what.

CHAPTER

THREE

WHAT MANIPULATORS ARE SUPPORTED?

The manipulators are divided into four groups:

- *foreground*, which changes text color;
- *background*, which changes text background color;
- *attributes*, which changes some text style (bold, underline, etc);
- *control*, which changes termcolor's behaviour.

Also, there are color manipulators for 16 colors, 256 colors and true colors palettes.

Note: While `termcolor` supports true color, it's required for the terminal emulator you use to run your software to support true color too. So please ensure it's supported before filing an issue.

3.1 Foreground manipulators

3.1.1 16 colors

1. `termcolor::grey`
2. `termcolor::red`
3. `termcolor::green`
4. `termcolor::yellow`
5. `termcolor::blue`
6. `termcolor::magenta`
7. `termcolor::cyan`
8. `termcolor::white`
9. `termcolor::bright_grey`
10. `termcolor::bright_red`
11. `termcolor::bright_green`
12. `termcolor::bright_yellow`
13. `termcolor::bright_blue`
14. `termcolor::bright_magenta`

15. termcolor::bright_cyan
16. termcolor::bright_white

3.1.2 256 colors

1. termcolor::color<256_COLOR_CODE>

3.1.3 true colors

1. termcolor::color<RED, GREEN, BLUE>

3.2 Background manipulators

3.2.1 16 colors

1. termcolor::on_grey
2. termcolor::on_red
3. termcolor::on_green
4. termcolor::on_yellow
5. termcolor::on_blue
6. termcolor::on_magenta
7. termcolor::on_cyan
8. termcolor::on_white
9. termcolor::on_bright_grey
10. termcolor::on_bright_red
11. termcolor::on_bright_green
12. termcolor::on_bright_yellow
13. termcolor::on_bright_blue
14. termcolor::on_bright_magenta
15. termcolor::on_bright_cyan
16. termcolor::on_bright_white

3.2.2 256 colors

1. `termcolor::on_color<256_COLOR_CODE>`

3.2.3 true colors

1. `termcolor::on_color<RED, GREEN, BLUE>`

3.3 Attribute manipulators

(Windows API does not support these manipulators except for `underline`)

1. `termcolor::bold`
2. `termcolor::dark`
3. `termcolor::italic`
4. `termcolor::underline`
5. `termcolor::blink`
6. `termcolor::reverse`
7. `termcolor::concealed`
8. `termcolor::crossed`

3.4 Control manipulators

(Windows API does not support these manipulators)

1. `termcolor::colorize`
2. `termcolor::nocolorize`

CHAPTER

FOUR

CAVEATS

1. On Windows, due to internal usage of <windows.h>, global namespace could be polluted with *min*/*max* macros. If such effect is desireable, please consider using `#define NOMINMAX` before `#include <termcolor.hpp>`.