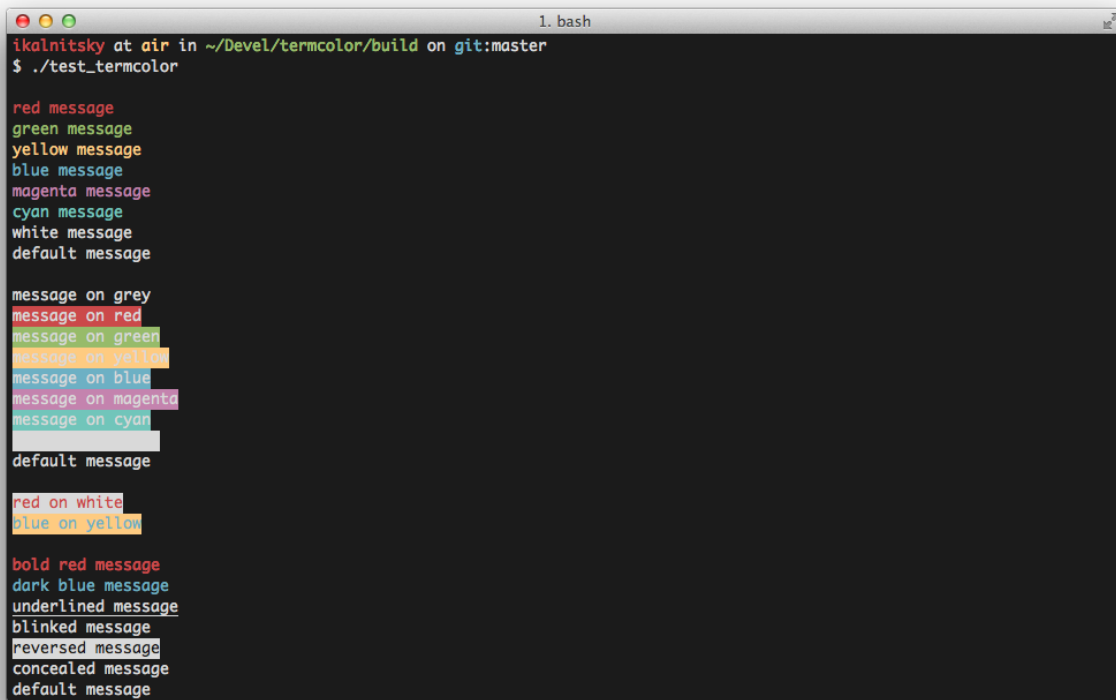

termcolor

Release 0.1

May 24, 2017

Contents

1	Installation	3
2	How to use?	5
3	What manipulators are supported?	7
3.1	Foreground manipulators	7
3.2	Background manipulators	7
3.3	Attribute manipulators	8

A terminal window titled "1. bash" showing the output of the command `./test_termcolor`. The output displays various color and formatting options for messages, including foreground colors (red, green, yellow, blue, magenta, cyan, white), background colors (grey, red, green, yellow, blue, magenta, cyan, white), and text styles (bold, dark blue, underlined, blinked, reversed, concealed).

```
ikalnitsky at air in ~/Devel/termcolor/build on git:master
$ ./test_termcolor

red message
green message
yellow message
blue message
magenta message
cyan message
white message
default message

message on grey
message on red
message on green
message on yellow
message on blue
message on magenta
message on cyan
message on white
default message

red on white
blue on yellow

bold red message
dark blue message
underlined message
blinked message
reversed message
concealed message
default message
```

Welcome to [termcolor](#)'s documentation. 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). On Windows, WinAPI is used instead, but some limitations are applied.

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

CHAPTER 1

Installation

Add `termcolor.hpp` to the project and use provided stream manipulators from the `termcolor` namespace.

CHAPTER 2

How to use?

It's very easy to use. The idea is based on the use of C++ stream manipulators. The typical «Hello World» application is below:

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

int main(int /*argc*/, char** /*argv*/)
{
    std::cout << termcolor::red << "Hello, Colorful World!" << std::endl;
    return 0;
}
```

The application above prints a string with red. It's obvious, isn't it? There is a one problem that is not obvious for the unexperienced users. If you write something this way:

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

the phrase «Here I'm» will be printed with red too. Why? Because you don't reset termcolor's setting. So if you want to print text with default terminal setting you have to reset termcolor's settings. It can be done by using termcolor::reset manipulator:

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

What manipulators are supported?

The manipulators are divided into three groups:

- *foreground*, which changes text color;
- *background*, which changes text background color;
- *attributes*, which changes some text style (bold, underline, etc).

Foreground manipulators

1. `termcolor::grey`
2. `termcolor::red`
3. `termcolor::green`
4. `termcolor::yellow`
5. `termcolor::blue`
6. `termcolor::magenta`
7. `termcolor::cyan`
8. `termcolor::white`

Background manipulators

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`

Attribute manipulators

(so far they don't supported by Windows)

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