

Font Effects

Code	Effect	Note
0	Reset / Normal	all attributes off
1	Bold or increased intensity	
2	Faint (decreased intensity)	Not widely supported.
3	Italic	Not widely supported. Sometimes treated as inverse.
4	Underline	
5	Slow Blink	less than 150 per minute
6	Rapid Blink	MS-DOS ANSI.SYS; 150+ per minute; not widely supported
7	[[reverse video]]	swap foreground and background colors
8	Conceal	Not widely supported.
9	Crossed-out	Characters legible, but marked for deletion. Not widely supported.
10	Primary(default) font	
11-19	Alternate font	Select alternate font n-10
20	Fraktur	hardly ever supported
21	Bold off or Double Underline	Bold off not widely supported; double underline hardly ever supported.
22	Normal color or intensity	Neither bold nor faint
23	Not italic, not Fraktur	
24	Underline off	Not singly or doubly underlined
25	Blink off	
27	Inverse off	
28	Reveal	conceal off
29	Not crossed out	
30-37	Set foreground color	See color table below
38	Set foreground color	Next arguments are 5;<n> or 2;<r>;<g>;, see below
39	Default foreground color	implementation defined (according to standard)
40-47	Set background color	See color table below
48	Set background color	Next arguments are 5;<n> or 2;<r>;<g>;, see below
49	Default background color	implementation defined (according to standard)
51	Framed	
52	Encircled	
53	Overlined	
54	Not framed or encircled	
55	Not overlined	
60	ideogram underline	hardly ever supported
61	ideogram double underline	hardly ever supported
62	ideogram overline	hardly ever supported
63	ideogram double overline	hardly ever supported
64	ideogram stress marking	hardly ever supported
65	ideogram attributes off	reset the effects of all of 60-64
90-97	Set bright foreground color	aixterm (not in standard)
100-107	Set bright background color	aixterm (not in standard)

2-bit Colours

You've got this already!

4-bit Colours

The standards implementing terminal colours began with limited (4-bit) options. The table below lists the RGB values of the background and foreground colours used for these by a variety of terminal emulators:

Using the above, you can make red text on a green background (but why?) using:

```
\033[31;42m
```

8-bit (256) colours

Technology advanced, and tables of 256 pre-selected colours became available, as shown below.

Using these above, you can make pink text like so:

```
\033[38;5;206m #That is, \033[38;5;<FG COLOR>m
```

And make an early-morning blue background using

```
\033[48;5;57m #That is, \033[48;5;<BG COLOR>m
```

And, of course, you can combine these:

```
\033[38;5;206;48;5;57m
```

The 8-bit colours are arranged like so:

Range	Description
-------	-------------

0x00-0x07	standard colors (same as the 4-bit colours)
-----------	---

0x08-0x0F	high intensity colors
-----------	-----------------------

0x10-0xE7	$6 \times 6 \times 6$ cube (216 colors): $16 + 36 \times r + 6 \times g + b$ ($0 \leq r, g, b \leq 5$)
-----------	--

0xE8-0xFF	grayscale from black to white in 24 steps
-----------	---

ALL THE COLOURS

Now we are living in the future, and the full RGB spectrum is available using:

```
\033[38;2;<r>;<g>;<b>m #Select RGB foreground color
```

```
\033[48;2;<r>;<g>;<b>m #Select RGB background color
```

So you can put pinkish text on a brownish background using

```
\033[38;2;255;82;197;48;2;155;106;0mHello
```

Support for "true color" terminals is listed [here](#).

A Handy Script to Remind Yourself


Since I'm often in the position of trying to remember what colours are what, I have a handy script called: `~/bin/ansi_colours`:

```
#!/usr/bin/env python3

for i in range(30, 37 + 1):
    print("\033[%dm%d\t\t\033[%dm%d" % (i, i, i + 60, i + 60))

print("\033[39m\033[49m          - Reset color")
print("\033[2K          - Clear Line")
print("\033[<L>;<C>H or \033[<L>;<C>f - Put the cursor at line L and column C.")
print("\033[<N>A          - Move the cursor up N lines")
print("\033[<N>B          - Move the cursor down N lines")
print("\033[<N>C          - Move the cursor forward N columns")
print("\033[<N>D          - Move the cursor backward N columns\n")
print("\033[2J          - Clear the screen, move to (0,0)")
print("\033[K          - Erase to end of line")
print("\033[s          - Save cursor position")
print("\033[u          - Restore cursor position\n")
print("\033[4m          - Underline on")
print("\033[24m          - Underline off\n")
print("\033[1m          - Bold on")
print("\033[21m          - Bold off")
```

This prints



31	90
32	91
33	92
34	93
35	94
36	95
37	96
	97