List of ANSI color escape sequences

Asked 12 years, 10 months ago Modified 4 months ago Viewed 460k times

486

On most terminals it is possible to colorize output using the `\033` ANSI escape sequence.

I'm looking for a list of all supported colors and options (like bright and blinking).

As there are probably differences between the terminals supporting them, I'm mainly interested in sequences supported by xterm-compatible terminals.

colors terminal ansi-escape

Share Follow

edited May 22, 2019 at 20:16
Dave Jarvis
30.7k 42 179 318

asked Jan 30, 2011 at 10:39
ThiefMaster
313k 84 595 637

8 Answers

Sorted by: Highest score (default)

1260

The ANSI escape sequences you're looking for are the Select Graphic Rendition subset. All of these have the form

```
\033[XXXm
```

where `xxx` is a series of semicolon-separated parameters.

To say, make text red, bold, and underlined (we'll discuss many other options below) in C you might write:

```
printf("\033[31;1;4mHello\033[0m");
```

In C++ you'd use

```
std::cout<<"\033[31;1;4mHello\033[0m";
```

In Python3 you'd use

```
print("\033[31;1;4mHello\033[0m")
```

and in Bash you'd use

```
echo -e "\033[31;1;4mHello\033[0m"
```

where the first part makes the text red (`31`), bold (`1`), underlined (`4`) and the last part clears all this (`0`).

As described in the table below, there are a large number of text properties you can set, such as boldness, font, underlining, &c.

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

Code	Effect	Note
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>;<b>` , 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>;<b>` , 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:

Name	FG Code	BG Code	VGA[nb 2]	Windows Console[nb 3]	Windows PowerShell[nb 4]	Windows 10 Console[nb 5] PowerShell 6	Terminal.app	PuTTY	mIRC	xterm	X[nb 6]	Ubuntu[nb 7]
Black	30	40	0,0,0			12,12,12	0,0,0					1,1,1
Red	31	41	170,0,0	128,0,0		197,15,31	194,54,33	187,0,0	127,0,0	205,0,0	255,0,0	222,56,43
Green	32	42	0,170,0	0,128,0		19,161,14	37,188,36	0,187,0	0,147,0	0,205,0	0,255,0	57,181,74
Yellow	33	43	170,85,0[nb 8]	128,128,0	238,237,240	193,156,0	173,173,39	187,187,0	252,127,0	205,205,0	255,255,0	255,199,6
Blue	34	44	0,0,170	0,0,128		0,55,218	73,46,225	0,0,187	0,0,127	0,0,238[23]	0,0,255	0,111,184
Magenta	35	45	170,0,170	128,0,128	1,36,86	136,23,152	211,56,211	187,0,187	156,0,156	205,0,205	255,0,255	118,38,113
Cyan	36	46	0,170,170	0,128,128		58,150,221	51,187,200	0,187,187	0,147,147	0,205,205	0,255,255	44,181,233
White	37	47	170,170,170	192,192,192		204,204,204	203,204,205	187,187,187	210,210,210	229,229,229	255,255,255	204,204,204
Bright Black	90	100	85,85,85	128,128,128		118,118,118	129,131,131	85,85,85	127,127,127	127,127,127		128,128,128
Bright Red	91	101	255,85,85	255,0,0		231,72,86	252,57,31	255,85,85	255,0,0	255,0,0		255,0,0
Bright Green	92	102	85,255,85	0,255,0		22,198,12	49,231,34	85,255,85	0,252,0	0,255,0	144,238,144	0,255,0
Bright Yellow	93	103	255,255,85	255,255,0		249,241,165	234,236,35	255,255,85	255,255,0	255,255,0	255,255,224	255,255,0
Bright Blue	94	104	85,85,255	0,0,255		59,120,255	88,51,255	85,85,255	0,0,252	92,92,255[24]	173,216,230	0,0,255
Bright Magenta	95	105	255,85,255	255,0,255		180,0,158	249,53,248	255,85,255	255,0,255	255,0,255		255,0,255
Bright Cyan	96	106	85,255,255	0,255,255		97,214,214	20,240,240	85,255,255	0,255,255	0,255,255	224,255,255	0,255,255
Bright White	97	107	255,255,255	255,255,255		242,242,242	233,235,235	255,255,255	255,255,255	255,255,255		255,255,255

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

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

11 Colours (An Interlude)

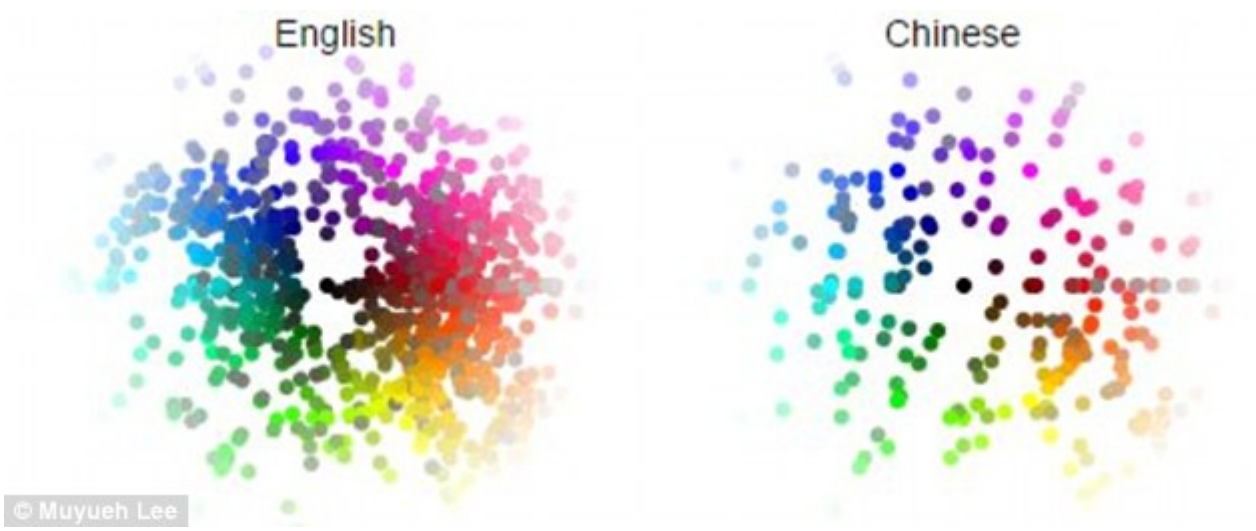
In their book "Basic Color Terms: Their Universality and Evolution", Brent Berlin and Paul Kay used data collected from twenty different languages from a range of language families to identify eleven possible basic color categories: white, black, red, green, yellow, blue, brown, purple, pink, orange, and gray.

Berlin and Kay found that, in languages with fewer than the maximum eleven color categories, the colors followed a specific evolutionary pattern. This pattern is as follows:

1. All languages contain terms for black (cool colours) and white (bright colours).
2. If a language contains three terms, then it contains a term for red.
3. If a language contains four terms, then it contains a term for either green or yellow (but not both).
4. If a language contains five terms, then it contains terms for both green and yellow.
5. If a language contains six terms, then it contains a term for blue.
6. If a language contains seven terms, then it contains a term for brown.
7. If a language contains eight or more terms, then it contains terms for purple, pink, orange or gray.

This may be why story *Beowulf* only contains the colours black, white, and red. It may also be why the *Bible* does not contain the colour blue. Homer's *Odyssey* contains black almost 200 times and white about 100 times. Red appears 15 times, while yellow and green appear only 10 times. (More information here)

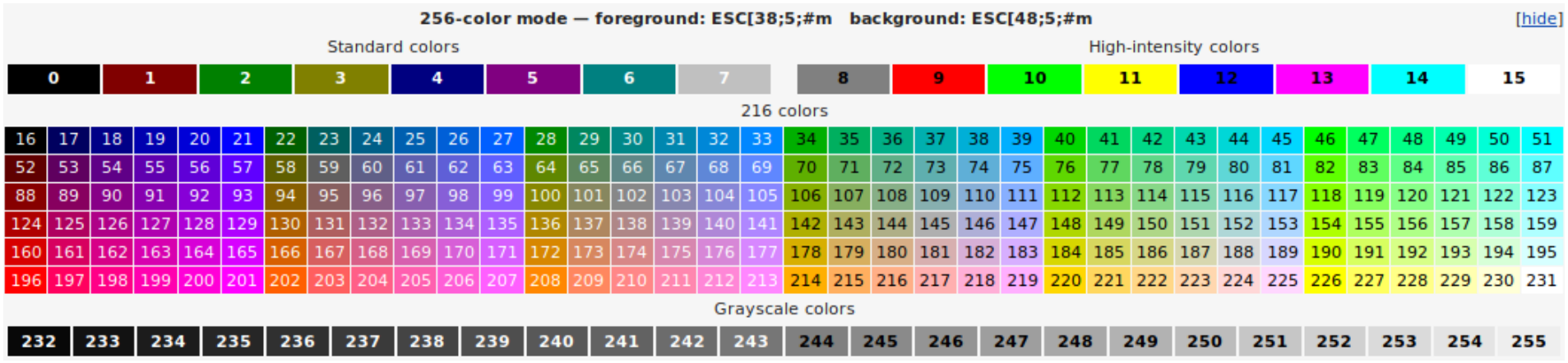
Differences between languages are also interesting: note the profusion of distinct colour words used by English vs. Chinese. However, digging deeper into these languages shows that each uses colour in distinct ways. (More information)



Generally speaking, the naming, use, and grouping of colours in human languages is fascinating. Now, back to the show.

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:

```
0x00-0x07:  standard colors (same as the 4-bit colours)
0x08-0x0F:  high intensity colors
0x10-0xE7:  6 × 6 × 6 cube (216 colors): 16 + 36 × r + 6 × g + b (0 ≤ r, g, b ≤ 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.

Much of the above is drawn from the Wikipedia page "ANSI escape code".

A Handy Script to Remind Yourself

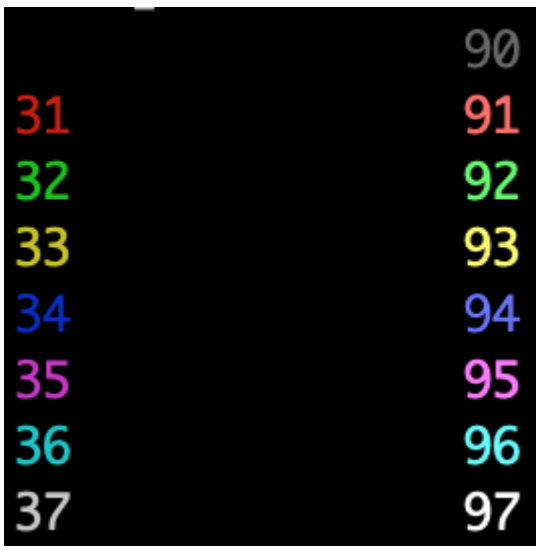
Since I'm often in the position of trying to remember what colours are what, I have a handy script called: `~/bin/ansi_colours`:

```
#!/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



Share Follow

edited Jul 18 at 4:22

answered Oct 19, 2015 at 4:49

Richard
57.5k 35 181 260

6 @giusti: Both `echo -e "\033[38;05;34;1mHi"` and `echo -e "\033[38;05;34m\033[1mHi"` worked for me, though anti-aliasing font effects did cause the appearance of the colour to change slightly under bolding in the terminal I was testing this on. – Richard Apr 12, 2018 at 16:36

6 The SGR (`\033[`) codes beginning with 38 and 48 *ought* to be separated with the otherwise reserved `:` as a sub-separator although this is not entirely clear from the primary sources at: ecma-international.org/publications/files/ECMA-ST/Ecma-048.pdf and itu.int/rec/... . Also some interpretations forget the color space Id in the `2` (16M-color RGB)/ `3` (16M-color CMY) / `4` (??? CMYK) forms - e.g. it should be `\033[38:2::255:255:255m` for a White 16M foreground and not `\033[38:2:255:255:255m` ! – SlySven Dec 19, 2018 at 23:37

3 The reason I go on about this is that a project Mudlet I code for has to handle both forms and I recently got up to my elbows in this to get it to work better... – SlySven May 6, 2019 at 16:42

3 Off topic (and 4 years later), but your interlude reminded me of this response (pfoley.public.iastate.edu/Decleapyear.htm) to a "bug" in VMS some time back. I hope you enjoy the read. – user3742898 May 16, 2019 at 18:12

24 I just wanted to find a list of ANSI colours and spent way too much time with reading articles on "basic colour terms". Thanks for the great distraction! :) – mzuther Jan 3, 2020 at 18:57

25

How about:

ECMA-48 - Control Functions for Coded Character Sets, 5th edition (June 1991) - A standard defining the color control codes, that is apparently supported also by xterm.

SGR 38 and 48 were originally reserved by ECMA-48, but were fleshed out a few years later in a joint ITU, IEC, and ISO standard, which comes in several parts and which (amongst a whole lot of other things) documents the SGR 38/48 control sequences for *direct colour* and *indexed colour*:

- *Information technology — Open Document Architecture (ODA) and interchange format: Document structures*. T.412. International Telecommunication Union.
- *Information technology — Open Document Architecture (ODA) and interchange format: Character content architectures*. T.416. International Telecommunication Union.
- *Information technology— Open Document Architecture (ODA) and Interchange Format: Character content architectures*. ISO/IEC 8613-6:1994. International Organization for Standardization.

There's a column for xterm in this table on the Wikipedia page for ANSI escape codes

Share Follow

edited May 14, 2018 at 20:11

JdeBP
2,130 16 24

answered Jan 30, 2011 at 10:43

sinelaw
16.3k 3 50 80

ECMA-48 does not define he ESC[38.. color control codes. – stevea Apr 24 at 7:05

22

When you write a ANSI escape code `\033[<color>m`, replace the `<color>` with any of the color codes below. For instance, `\033[31m` would be red text color:

Color	Example	Text	Background	Bright Text	Bright Background
Black		30	40	90	100
Red		31	41	91	101
Green		32	42	92	102

Color	Example	Text	Background	Bright Text	Bright Background
Yellow		33	43	93	103
Blue		34	44	94	104
Magenta		35	45	95	105
Cyan		36	46	96	106
White		37	47	97	107
Default		39	49	99	109

Also, remember to use `\033[0m` every time you want to revert back to the default terminal text style. Otherwise, any color or styling may spill over and into other terminal messages.

For effects, the codes are:

Effect	On	Off	Example
Bold	1	21	This is BOLD!
Dim	2	22	This is DIMMED!
Underline	4	24	
Blink	5	25	
Reverse	7	27	
Hide	8	28	

I recommend these articles to explore further:

- https://notes.burke.libbey.me/ansi-escape-codes/
- https://www.lihaoyi.com/post/BuildyourownCommandLinewithANSIescapecodes.html

PS: In full disclosure, I'm the author of the Colorist package. Colorist is lightweight and makes it easy to print colorful text in many terminals. Simply install the package with `pip install colorist` and type:

```
from colorist import Color

print(f"Only {Color.CYAN}this part{Color.OFF} is in colour")
```

```
Only this part is in colour
```

Moreover, Colorist also supports color defined as RGB, HSL or Hex if your terminal supports advanced ANSI colors:

```
from colorist import ColorRGB, BgColorRGB

dusty_pink = ColorRGB(194, 145, 164)
bg_steel_blue = BgColorRGB(70, 130, 180)

print(f"I want to use {dusty_pink}dusty pink{dusty_pink.OFF} and
{bg_steel_blue}steel blue{bg_steel_blue.OFF} colors inside this paragraph")
```

```
I want to use dusty pink and steel blue colors inside this paragraph
```

```
from colorist import ColorHSL, BgColorHSL

mustard_green = ColorHSL(60, 56, 43)
bg_steel_gray = BgColorHSL(190, 2, 49)

print(f"I want to use {mustard_green}mustard green{mustard_green.OFF} and
{bg_steel_gray}steel blue{bg_steel_gray.OFF} colors inside this paragraph")
```

```
I want to use mustard green and steel blue colors inside this paragraph
```

```
from colorist import ColorHex, BgColorHex
```

```
watermelon_red = ColorHex("#ff5733")
bg_mint_green = BgColorHex("#99ff99")

print(f"I want to use {watermelon_red}watermelon pink{watermelon_red.OFF} and
{bg_mint_green}mint green{bg_mint_green.OFF} colors inside this paragraph")
```

```
I want to use watermelon pink and mint green colors inside this paragraph
```

More options with [Colorist](#):

Foreground Text

Color	Full Text Function	Custom	Example
	`green("text")`	`Color.GREEN`	This is GREEN!
	`yellow("text")`	`Color.YELLOW`	This is YELLOW!
	`red("text")`	`Color.RED`	This is RED!
	`magenta("text")`	`Color.MAGENTA`	This is MAGENTA!
	`blue("text")`	`Color.BLUE`	This is BLUE!
	`cyan("text")`	`Color.CYAN`	This is CYAN!
	`white("text")`	`Color.WHITE`	This is WHITE!
	`black("text")`	`Color.BLACK`	This is BLACK!
-	-	`Color.OFF`	-
	`bright_green("text")`	`BrightColor.GREEN`	This is BRIGHT GREEN!
	`bright_yellow("text")`	`BrightColor.YELLOW`	This is BRIGHT YELLOW!
	`bright_red("text")`	`BrightColor.RED`	This is BRIGHT RED!
	`bright_magenta("text")`	`BrightColor.MAGENTA`	This is BRIGHT MAGENTA!
	`bright_blue("text")`	`BrightColor.BLUE`	This is BRIGHT BLUE!
	`bright_cyan("text")`	`BrightColor.CYAN`	This is BRIGHT CYAN!
	`bright_white("text")`	`BrightColor.WHITE`	This is BRIGHT WHITE!
	`bright_black("text")`	`BrightColor.BLACK`	This is BRIGHT BLACK!
-	-	`BrightColor.OFF`	-

Background

Color	Full Text Function	Custom	Example
	`bg_green("text")`	`BgColor.GREEN`	This is GREEN background!
	`bg_yellow("text")`	`BgColor.YELLOW`	This is YELLOW background!
	`bg_red("text")`	`BgColor.RED`	This is RED background!
	`bg_magenta("text")`	`BgColor.MAGENTA`	This is MAGENTA background!
	`bg_blue("text")`	`BgColor.BLUE`	This is BLUE background!
	`bg_cyan("text")`	`BgColor.CYAN`	This is CYAN background!
	`bg_white("text")`	`BgColor.WHITE`	This is WHITE background!
	`bg_black("text")`	`BgColor.BLACK`	This is BLACK background!
-	-	`BgColor.OFF`	-
	`bg_bright_green("text")`	`BgBrightColor.GREEN`	This is GREEN background!
	`bg_bright_yellow("text")`	`BgBrightColor.YELLOW`	This is YELLOW background!
	`bg_bright_red("text")`	`BgBrightColor.RED`	This is RED background!
	`bg_bright_magenta("text")`	`BgBrightColor.MAGENTA`	This is MAGENTA background!
	`bg_bright_blue("text")`	`BgBrightColor.BLUE`	This is BLUE background!
	`bg_bright_cyan("text")`	`BgBrightColor.CYAN`	This is CYAN background!
	`bg_bright_white("text")`	`BgBrightColor.WHITE`	This is WHITE background!
	`bg_bright_black("text")`	`BgBrightColor.BLACK`	This is BLACK background!
-	-	`BgBrightColor.OFF`	-

Effects

Effect	Full Text Function	Custom	Reset	Example
Bold	`effect_bold("text")`	`Effect.BOLD`	`Effect.BOLD_OFF`	This is BOLD!
Dim	`effect_dim("text")`	`Effect.DIM`	`Effect.DIM_OFF`	This is DIMMED!
Underline	`effect_underline("text")`	`Effect.UNDERLINE`	`Effect.UNDERLINE_OFF`	This is UNDERLINED!
Blink	`effect_blink("text")`	`Effect.BLINK`	`Effect.BLINK_OFF`	This is BLINKING!
Reverse	`effect_reverse("text")`	`Effect.REVERSE`	`Effect.REVERSE_OFF`	This is REVERSED!
Hide	`effect_hide("text")`	`Effect.HIDE`	`Effect.HIDE_OFF`	
-	-	-	`Effect.OFF`	-

Share Follow

edited Aug 13 at 9:19

answered Apr 11 at 12:19

Jakob Bagterp

680 5 12

13

There are some more interesting ones along with related info.

- ~~http://wiki.bash-hackers.org/scripting/terminalcodes~~ (dead; archive.org spanshot)
- ~~http://www.termsys.demon.co.uk/vtansi.htm~~ (dead; archive.org snapshot)
- http://invisible-island.net/xterm/ctlseqs/ctlseqs.html
- http://www.tldp.org/HOWTO/Bash-Prompt-HOWTO/c327.html
- https://wiki.archlinux.org/index.php/Color_Bash_Prompt

Share Follow

edited Jun 9 at 10:07

answered Dec 11, 2013 at 10:39
Palec
12.9k 8 69 138

Um, I made an edit to repair a dead-link but I forgot I wasn't signed in at the time and I messed up the formatting slightly (I was trying to leave the old link but have it crossed-out) - unfortunately I cannot get at the edit to revise it, even though I am now signed in... 8-P – SlySven Jul 15, 2020 at 3:38

3 Thanks for the edit, I fixed it. Sadly, links die and link-based answers like this one are bound to rot. Would not create answers like this anymore -- but keeping this one as it still mostly works as expected. :-) – Palec Jul 20, 2020 at 12:06

6

For these who don't get proper results other than mentioned languages, if you're using C# to print a text into console(terminal) window you should replace **"\033"** with **"\x1b"**. In Visual Basic it would be **Chrw(27)**.

Share Follow

answered May 23, 2019 at 14:32
HolyRandom
47 1 13

In Java: `(char) 27` or `"\u001b"` – Jesse Glick Oct 13, 2022 at 21:31

2 `\033` or just `\33` works fine in Java, the same way as in many other languages. – Holger Nov 4, 2022 at 16:07

5

It's related absolutely to your terminal. VTE doesn't support blink, If you use `gnome-terminal` , `tilda` , `guake` , `terminator` , `xfce4-terminal` and so on according to VTE, you won't have blink.
If you use or want to use blink on VTE, you have to use `xterm` .
You can use infocmp command with terminal name:

```
#infocmp vt100
#infocmp xterm
#infocmp vte
```

For example :

```
# infocmp vte
#   Reconstructed via infocmp from file: /usr/share/terminfo/v/vte
vte|VTE aka GNOME Terminal,
    am, bce, mir, msgr, xenl,
    colors#8, cols#80, it#8, lines#24, ncv#16, pairs#64,
    acsc=``aaffggiijjkkllmmnnooppqqrrssttuuvvwwxxyyzz{{||}}~~,
    bel=^G, bold=\E[1m, civis=\E[?25l, clear=\E[H\E[2J,
    cnorm=\E[?25h, cr=^M, csr=\E[%i%p1%d;%p2%dr,
    cub=\E[%p1%dD, cub1=^H, cud=\E[%p1%dB, cud1=^J,
    cuf=\E[%p1%dC, cuf1=\E[C, cup=\E[%i%p1%d;%p2%dH,
    cuu=\E[%p1%dA, cuu1=\E[A, dch=\E[%p1%dP, dch1=\E[P,
    dim=\E[2m, dl=\E[%p1%dM, dl1=\E[M, ech=\E[%p1%dX, ed=\E[J,
    el=\E[K, enacs=\E)0, home=\E[H, hpa=\E[%i%p1%dG, ht=^I,
    hts=\EH, il=\E[%p1%dL, il1=\E[L, ind=^J, invis=\E[8m,
    is2=\E[m\E[?7h\E[4l\E>\E7\E[r\E[?1;3;4;6l\E8,
    kDC=\E[3;2~, kEND=\E[1;2F, kHOM=\E[1;2H, kIC=\E[2;2~,
    kLFT=\E[1;2D, kNXT=\E[6;2~, kPRV=\E[5;2~, kRIT=\E[1;2C,
    kb2=\E[E, kbs=\177, kcbt=\E[Z, kcub1=\EOD, kcud1=\EOB,
    kcuf1=\EOC, kcuu1=\EOA, kdch1=\E[3~, kend=\EOF, kf1=\EOP,
    kf10=\E[21~, kf11=\E[23~, kf12=\E[24~, kf13=\E[1;2P,
    kf14=\E[1;2Q, kf15=\E[1;2R, kf16=\E[1;2S, kf17=\E[15;2~,
    kf18=\E[17;2~, kf19=\E[18;2~, kf2=\EOQ, kf20=\E[19;2~,
    kf21=\E[20;2~, kf22=\E[21;2~, kf23=\E[23;2~,
    kf24=\E[24;2~, kf25=\E[1;5P, kf26=\E[1;5Q, kf27=\E[1;5R,
    kf28=\E[1;5S, kf29=\E[15;5~, kf3=\EOR, kf30=\E[17;5~,
    kf31=\E[18;5~, kf32=\E[19;5~, kf33=\E[20;5~,
    kf34=\E[21;5~, kf35=\E[23;5~, kf36=\E[24;5~,
    kf37=\E[1;6P, kf38=\E[1;6Q, kf39=\E[1;6R, kf4=\EOS,
    kf40=\E[1;6S, kf41=\E[15;6~, kf42=\E[17;6~,
    kf43=\E[18;6~, kf44=\E[19;6~, kf45=\E[20;6~,
    kf46=\E[21;6~, kf47=\E[23;6~, kf48=\E[24;6~,
    kf49=\E[1;3P, kf5=\E[15~, kf50=\E[1;3Q, kf51=\E[1;3R,
```

```
kf52=\E[1;3S, kf53=\E[15;3~, kf54=\E[17;3~,
kf55=\E[18;3~, kf56=\E[19;3~, kf57=\E[20;3~,
kf58=\E[21;3~, kf59=\E[23;3~, kf6=\E[17~, kf60=\E[24;3~,
kf61=\E[1;4P, kf62=\E[1;4Q, kf63=\E[1;4R, kf7=\E[18~,
kf8=\E[19~, kf9=\E[20~, kfnd=\E[1~, khome=\EOH,
```

Share Follow

edited Oct 5, 2020 at 13:08
Community Bot
1 1

answered Mar 16, 2016 at 20:03
PersianGulf
2,855 6 48 67

VTE 0.52 / gnome-terminal 3.28 adds support for blinking text (and so it will work in other VTE-based emulators too). – egmont Mar 11, 2018 at 21:33

2

Here is some code that shows all escape sequences that have to do with color. You might need to get the actual escape character in order for the code to work.

```
@echo off
:top
cls
echo [101;93m STYLES [0m
echo ^<ESC^>[0m [0mReset[0m
echo ^<ESC^>[1m [1mBold[0m
echo ^<ESC^>[4m [4mUnderline[0m
echo ^<ESC^>[7m [7mInverse[0m
echo.
echo [101;93m NORMAL FOREGROUND COLORS [0m
echo ^<ESC^>[30m [30mBlack[0m (black)
echo ^<ESC^>[31m [31mRed[0m
echo ^<ESC^>[32m [32mGreen[0m
echo ^<ESC^>[33m [33mYellow[0m
echo ^<ESC^>[34m [34mBlue[0m
echo ^<ESC^>[35m [35mMagenta[0m
echo ^<ESC^>[36m [36mCyan[0m
echo ^<ESC^>[37m [37mWhite[0m
echo.
echo [101;93m NORMAL BACKGROUND COLORS [0m
echo ^<ESC^>[40m [40mBlack[0m
echo ^<ESC^>[41m [41mRed[0m
echo ^<ESC^>[42m [42mGreen[0m
echo ^<ESC^>[43m [43mYellow[0m
echo ^<ESC^>[44m [44mBlue[0m
echo ^<ESC^>[45m [45mMagenta[0m
echo ^<ESC^>[46m [46mCyan[0m
echo ^<ESC^>[47m [47mWhite[0m (white)
echo.
echo [101;93m STRONG FOREGROUND COLORS [0m
echo ^<ESC^>[90m [90mWhite[0m
echo ^<ESC^>[91m [91mRed[0m
echo ^<ESC^>[92m [92mGreen[0m
echo ^<ESC^>[93m [93mYellow[0m
echo ^<ESC^>[94m [94mBlue[0m
echo ^<ESC^>[95m [95mMagenta[0m
echo ^<ESC^>[96m [96mCyan[0m
echo ^<ESC^>[97m [97mWhite[0m
```

Share Follow

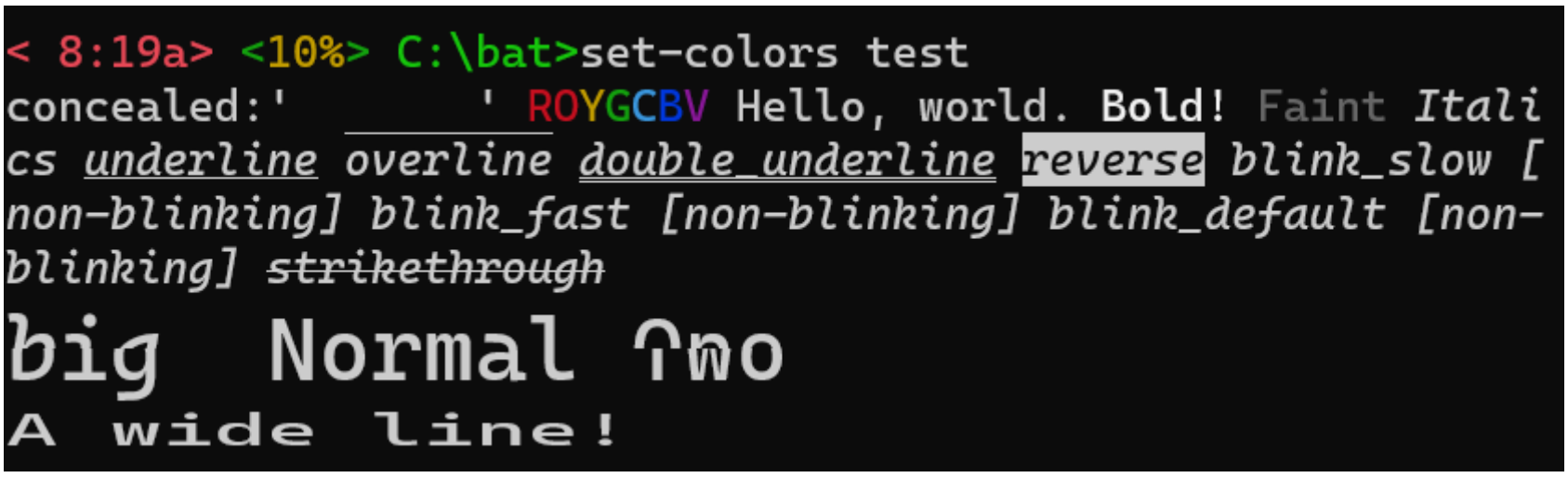
answered Jul 23, 2022 at 2:31
NotePro.bat
66 5

Is this a telnet script or something? – arcanemachine Apr 12 at 2:28

1 @arcanemachine that's batch (windows scripting language) – mazunki Sep 12 at 12:32

0

If you're using TCC shell (and this only requires modifying a line or two to work with CMD, since i use %@CHAR, which is TCC-specific), here's a handy script that lets you test most ansi via convenient environment variables. Here's my results with Windows Terminal, which supports a lot, but not all, of this, including double-height and wide lines:



```
rem ANSI: Initialization
        rem set up basic beginning of all ansi codes
            set ESCAPE=%@CHAR[27]
            set ANSI_ESCAPE=%@CHAR[27][
                set ANSIESCAPE=%ANSI_ESCAPE%

rem ANSI: special stuff: reset
            set ANSI_RESET=%ANSI_ESCAPE%0m
                set ANSIRESET=%ANSI_RESET%

rem ANSI: special stuff: position save/restore
            set ANSI_POSITION_SAVE=%ESCAPE%7%ANSI_ESCAPE%s                  %+ REM we do this the DEC way, then the SCO way
            set ANSI_POSITION_RESTORE=%ESCAPE%8%ANSI_ESCAPE%u               %+ REM we do this the DEC way, then the SCO way
                set ANSI_SAVE_POSITION=%ANSI_POSITION_SAVE%
                set ANSI_RESTORE_POSITION=%ANSI_POSITION_RESTORE%
            set ANSI_POSITION_REQUEST=%ANSI_ESCAPE%6n                       %+ REM request cursor position (reports as
ESC[#;#R)
                set ANSI_REQUEST_POSITION=%ANSI_POSITION_REQUEST%

rem ANSI: position movement
        rem To Home
            set ANSI_HOME=%ANSI_ESCAPE%H                                    %+ REM moves cursor to home position (0, 0)
                set ANSI_MOVE_HOME=%ANSI_HOME%
                set ANSI_MOVE_TO_HOME=%ANSI_HOME%

        rem To a specific position
            function ANSI_MOVE_TO_POS1=`%@CHAR[27][%1;%2H`                  %+ rem moves cursor to line #, column #\_____ both
work
            function ANSI_MOVE_TO_POS2=`%@CHAR[27][%1;%2f`                  %+ rem moves cursor to line #, column #/
                function ANSI_MOVE_POS=`%@CHAR[27][%1;%2H`                  %+ rem alias
                function ANSI_MOVE=`%@CHAR[27][%1;%2H`                      %+ rem alias
            function ANSI_MOVE_TO_COL=`%@CHAR[27][%1G`                      %+ rem moves cursor to column #
            function ANSI_MOVE_TO_ROW=`%@CHAR[27][%1H`                      %+ rem unfortunately does not preserve column
position! not possible! cursor request ansi code return value cannot be captured

        rem Up/Down/Left/Right
            set ANSI_MOVE_UP_1=%ESCAPE%M                                    %+ rem moves cursor one line up, scrolling if
```

Share Follow

answered Jul 15 at 12:23

ClioCJS

64 3 11