**What color codes can I use in my bash PS1 prompt?**

I used several colors in my bash PS1 prompt such as

\033]01;31\] # pink

\033]00m\] # white

\033]01;36\] # bold green

\033]02;36\] # green

\033]01;34\] # blue

\033]01;33\] # bold yellow

Where can I find a list of the color codes I can use?

I looked at [Colorize Bash Console Color](https://unix.stackexchange.com/questions/74024/colorize-bash-console-color) but it didn't answer my question about a list of the actual codes.

It would be nice if there was a more readable form also.

See also <https://unix.stackexchange.com/a/127800/10043>

244

Those are [ANSI escape sequences](http://en.wikipedia.org/wiki/ANSI_escape_code#Colors); that link is to a chart of color codes but there are other interesting things on that Wikipedia page as well. Not all of them work on (e.g.) a normal Linux console.

This is incorrect:

\033]00m\] # white

0 resets the terminal to its default (which is probably white). The actual code for white foreground is 37. Also, the escaped closing brace at the end (\]) is not part of the color sequence (see the last few paragraphs below for an explanation of their purpose in setting a prompt).

Note that some GUI terminals allow you to specify a customized color scheme. This will affect the output.

There's [a list here](http://misc.flogisoft.com/bash/tip_colors_and_formatting) which adds 7 foreground and 7 background colors I had not seen before, but they seem to work:

# Foreground colors

90 Dark gray

91 Light red

92 Light green

93 Light yellow

94 Light blue

95 Light magenta

96 Light cyan

# Background colors

100 Dark gray

101 Light red

102 Light green

103 Light yellow

104 Light blue

105 Light magenta

106 Light cyan

In addition, if you have a 256 color GUI terminal (I think most of them are now), you can apply colors from this chart:



The ANSI sequence to select these, using the number in the bottom left corner, starts 38;5; for the foreground and 48;5; for the background, then the color number, so e.g.:

echo -e "\\033[48;5;95;38;5;214mhello world\\033[0m"

Gives me a light orange on tan (meaning, the color chart is roughly approximated).

You can see the colors in this chart1 as they would appear on your terminal fairly easily:

#!/bin/bash

color=16;

while [ $color -lt 245 ]; do

echo -e "$color: \\033[38;5;${color}mhello\\033[48;5;${color}mworld\\033[0m"

((color++));

done

The output is self-explanatory.

Some systems set the $TERM variable to xterm-256color if you are on a 256 color terminal via some shell code in /etc/profile. On others, you should be able to configure your terminal to use this. That will let TUI applications know there are 256 colors, and allow you to add something like this to your ~/.bashrc:

if [[ "$TERM" =~ 256color ]]; then

PS1="MyCrazyPrompt..."

fi

Beware that when you use color escape sequences in your prompt, you should enclose them in escaped (\ prefixed) square brackets, like this:

PS1="\[\033[01;32m\]MyPrompt: \[\033[0m\]"

Notice the ['s interior to the color sequence are not escaped, but the enclosing ones are. The purpose of the latter is to indicate to the shell that the enclosed sequence does not count toward the character length of the prompt. If that count is wrong, weird things will happen when you scroll back through the history, e.g., if it is too long, the excess length of the last scrolled string will appear attached to your prompt and you won't be able to backspace into it (it's ignored the same way the prompt is).

Also note that if you want to include the output of a command run every time the prompt is used (as opposed to just once when the prompt is set), you should set it as a literal string with single quotes, e.g.:

PS1='\[\033[01;32m\]$(date): \[\033[0m\]'

Although this is not a great example if you are happy with using bash's special \d or \D{format} prompt escapes -- which are not the topic of the question but can be found in man bash under PROMPTING. There are various other useful escapes such as \w for current directory, \u for current user, etc.

1. The main portion of this chart, colors 16 - 231 (notice they are not in numerical order) are a 6 x 6 x 6 RGB color cube. "Color cube" refers to the fact that an RGB color space can be represented using a three dimensional array (with one axis for red, one for green, and one for blue). Each color in the cube here can be represented as coordinates in a 6 x 6 x 6 array, and the index in the chart calculated thusly:

16 + R \* 36 + G \* 6 + B

The first color in the cube, at index 16 in the chart, is black (RGB 0, 0, 0). You could use this formula in shell script:

#!/bin/sh

function RGBcolor {

echo "16 + $1 \* 36 + $2 \* 6 + $3" | bc

}

fg=$(RGBcolor 1 0 2) # Violet

bg=$(RGBcolor 5 3 0) # Bright orange.

echo -e "\\033[1;38;5;$fg;48;5;${bg}mviolet on tangerine\\033[0m"

Looks like at least some of the list is:

txtblk='\e[0;30m' # Black - Regular

txtred='\e[0;31m' # Red

txtgrn='\e[0;32m' # Green

txtylw='\e[0;33m' # Yellow

txtblu='\e[0;34m' # Blue

txtpur='\e[0;35m' # Purple

txtcyn='\e[0;36m' # Cyan

txtwht='\e[0;37m' # White

bldblk='\e[1;30m' # Black - Bold

bldred='\e[1;31m' # Red

bldgrn='\e[1;32m' # Green

bldylw='\e[1;33m' # Yellow

bldblu='\e[1;34m' # Blue

bldpur='\e[1;35m' # Purple

bldcyn='\e[1;36m' # Cyan

bldwht='\e[1;37m' # White

unkblk='\e[4;30m' # Black - Underline

undred='\e[4;31m' # Red

undgrn='\e[4;32m' # Green

undylw='\e[4;33m' # Yellow

undblu='\e[4;34m' # Blue

undpur='\e[4;35m' # Purple

undcyn='\e[4;36m' # Cyan

undwht='\e[4;37m' # White

bakblk='\e[40m' # Black - Background

bakred='\e[41m' # Red

bakgrn='\e[42m' # Green

bakylw='\e[43m' # Yellow

bakblu='\e[44m' # Blue

bakpur='\e[45m' # Purple

bakcyn='\e[46m' # Cyan

bakwht='\e[47m' # White

txtrst='\e[0m' # Text Reset

based on <https://wiki.archlinux.org/index.php/Color_Bash_Prompt>

I wrote a bash function that can show you all the colors, if this helps.

function colorgrid( )

{

iter=16

while [ $iter -lt 52 ]

do

second=$[$iter+36]

third=$[$second+36]

four=$[$third+36]

five=$[$four+36]

six=$[$five+36]

seven=$[$six+36]

if [ $seven -gt 250 ];then seven=$[$seven-251]; fi

echo -en "\033[38;5;$(echo $iter)m█ "

printf "%03d" $iter

echo -en " \033[38;5;$(echo $second)m█ "

printf "%03d" $second

echo -en " \033[38;5;$(echo $third)m█ "

printf "%03d" $third

echo -en " \033[38;5;$(echo $four)m█ "

printf "%03d" $four

echo -en " \033[38;5;$(echo $five)m█ "

printf "%03d" $five

echo -en " \033[38;5;$(echo $six)m█ "

printf "%03d" $six

echo -en " \033[38;5;$(echo $seven)m█ "

printf "%03d" $seven

iter=$[$iter+1]

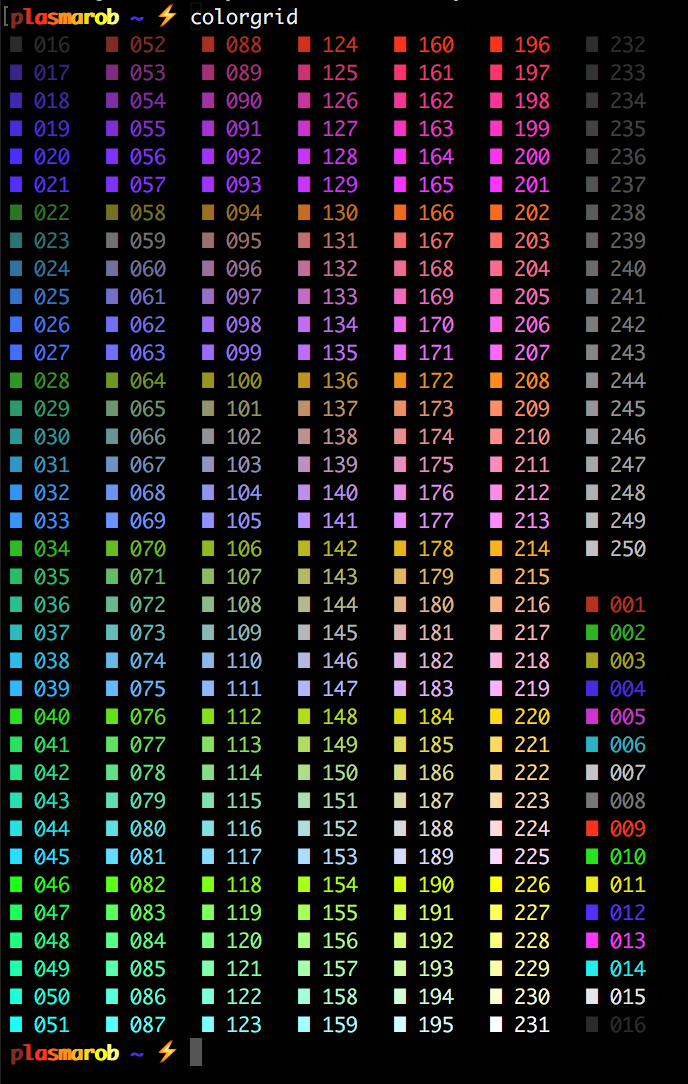
printf '\r\n'

done

}

You can throw that in a .bashrc / .bash\_profile / .bash\_aliases or save it as a script and run it that way. You can use the colors to change color like I did with my name below.

colorgrid() outputs:



I changed my name in my .bash\_profile by doing this:

if [ "$USER" = "plasmarob" ]; then

p="\[\033[01;38;5;52m\]p"

l="\[\033[01;38;5;124m\]l"

a="\[\033[01;38;5;196m\]a"

s="\[\033[01;38;5;202m\]s"

m="\[\033[01;38;5;208m\]m"

a2="\[\033[01;38;5;214m\]a"

r="\[\033[01;38;5;220m\]r"

o="\[\033[01;38;5;226m\]o"

b="\[\033[01;38;5;228m\]b"

local \_\_user\_and\_host="$p$l$a$s$m$a2$r$o$b"

else

local \_\_user\_and\_host="\[\033[01;36m\]\u"

fi

...

export PS1="$\_\_user\_and\_host $\_\_cur\_location $\_\_git\_branch\_color$\_\_git\_branch$\_\_prompt\_tail$\_\_last\_color "

Note that the 01 prefix in a string like \[\033[01;38;5;214m\]a sets it to be bold.

12

Another script like the one posted by TAFKA 'goldilocks' for displaying colors which is maybe a little more practical for reference purposes:

#!/bin/bash

useage() {

printf "\n\e[1;4mAscii Escape Code Helper Utility\e[m\n\n"

printf " \e[1mUseage:\e[m colors.sh [-|-b|-f|-bq|-fq|-?|?] [start] [end] [step]\n\n"

printf "The values for the first parameter may be one of the following:\n\n"

printf " \e[1m-\e[m Will result in the default output.\n"

printf " \e[1m-b\e[m This will display the 8 color version of this chart.\n"

printf " \e[1m-f\e[m This will display the 256 color version of this chart using foreground colors.\n"

printf " \e[1m-q\e[m This will display the 256 color version of this chart without the extra text.\n"

printf " \e[1m-bq\e[m This will display the 8 color version of this chart without the extra text.\n"

printf " \e[1m-fq\e[m This will display the 256 color version of this chart using foreground colors without the extra text.\n"

printf " \e[1m-?|?\e[m Displays this help screen.\n"

printf "\nThe remaining parameters are only used if the first parameter is one of: \e[1m-,-f,q,fq\e[m\n\n"

printf " \e[1mstart\e[m The color index to begin display at.\n"

printf " \e[1mend\e[m The color index to stop display at.\n"

printf " \e[1mstart\e[m The number of indexes to increment color by each iteration.\n\n\n"

}

verbose() {

if [[ "$1" != "-q" && "$1" != "-fq" && "$1" != "-bq" ]]; then

printf "\nTo control the display style use \e[1m%s\e[m where \e[1m%s\e[m is:\n" '\e[{$value}[:{$value}]m' '{$value}'

printf "\n 0 Normal \e[1m1 Bold\e[m \e[2m2 Dim\e[m \e[3m3 ???\e[m \e[4m4 Underlined\e[m \e[5m5 Blink\e[m \e[6m6 ???\e[m \e[7m7 Inverted\e[m \e[8m8 Hidden\e[m\n\n"

printf "If \e[1m%s\e[m is not provided it will reset the display.\n\n" '{$value}'

fi

}

eight\_color() {

local fgc bgc vals seq0

if [ "$1" != "-bq" ]; then

printf "\n\e[1;4m8 Color Escape Value Pallette\e[m\n\n"

printf "Color escapes are \e[1m%s\e[m\n" '\e[${value};...;${value}m'

printf " Values \e[1m30..37\e[m are \e[1mforeground\e[m colors\n"

printf " Values \e[1m40..47\e[m are \e[1mbackground\e[m colors\n\n"

fi

for fgc in {30..37}; do

for bgc in {40..47}; do

fgc=${fgc#37}

bgc=${bgc#40}

vals="${fgc:+$fgc;}${bgc}"

vals=${vals%%;}

seq0="${vals:+\e[${vals}m}"

printf " %-9s" "${seq0:-(default)}"

printf " ${seq0}TEXT\e[m"

printf " \e[${vals:+${vals+$vals;}}1mBOLD\e[m"

done

printf "\e[0m\n"

done

}

if [[ "$1" == "-b" || "$1" == "-bq" ]]; then

eight\_color "$1"

verbose "$1"

elif [[ "$1" == "" || "$1" == "-" || "$1" == "-f" || "$1" == "-q" || "$1" == "-fq" ]]; then

start=${2:-0}

end=${3:-255}

step=${4:-1}

color=$start

style="48;5;"

if [[ "$1" == "-f" || "$1" == "-fq" ]]; then

style="38;5;"

fi

perLine=$(( ( $(tput cols) - 2 ) / 9 ));

if [[ "$1" != "-q" && "$1" != "-fq" ]]; then

printf "\n\e[1;4m256 Color Escape Value Pallette\e[0m\n\n"

printf " \e[1m%s\e[m for \e[1mbackground\e[m colors\n \e[1m%s\e[m for \e[1mforeground\e[m colors\n\n" '\e[48;5;${value}m' '\e[38;5;${value}m'

fi

while [ $color -le $end ]; do

printf "\e[m \e[${style}${color}m %3d \e[m " $color

((color+=step))

if [ $(( ( ( $color - $start ) / $step ) % $perLine )) -eq 0 ]; then

printf "\n"

fi

done

printf "\e[m\n"

verbose "$1"

else

useage

fi

This should size correctly for the terminal you are using. It is a little over the top for this purpose but now you can control many aspects of how this displays via parameters. Hopefully, they are all self explanatory.

0

cat "$0" 1>&2;

#

# = 256-color test =

#

# [

# |\*| Source: https://unix.stackexchange.com/a/643715

# |\*| Source (original): https://misc.flogisoft.com/bash/tip\_colors\_and\_formatting#colors2

# |\*| Last update: CE 2021-05-15 03:17 UTC ]

#

#

# This script shall echo a bunch of color codes generating a fancy color table: demonstrating the 256-color compatibility of the shell / terminal.

#

#

#

#

# == Implementation ==

#

# === Table 0..15 ===

Colors="$(

# Colors (0..15):

i=0;

while

echo "$i";

[ $i -lt 15 ];

do

i=$(( $i + 1 ));

done;

)";

echo;

for x0 in \

'48' '38'; # Background / Foreground

do {

for Color in \

$Colors;

do {

printf '\e['"$x0"';5;%sm %3s ' \

"$Color" "$Color"; # [Note 1]

# 8 entries per line:

[ $(( ($Color + 1) % 8 )) -eq 0 ] && echo -e '\e[m'; # [Note 1]

};

done;

};

done;

unset Colors x0;

echo;

# === Table 16..255 ===

for Color in \

$(

# Colors (16..255):

i=16;

while

echo "$i";

[ $i -lt 255 ];

do

i=$(( $i + 1 ));

done;

);

do {

Colors="$Colors $Color";

# 6 entries per group:

[ $(( ($Color + 1) % 6 )) -eq 4 ] && {

for x0 in \

'38' '48'; # Foreground / Background

do {

for Color in \

$Colors;

do

printf '\e['"$x0"';5;%sm %3s ' \

"$Color" "$Color"; # [Note 1]

done;

echo -ne '\e[m'; # [Note 1]

};

done;

unset Colors x0;

echo;

};

};

done;

unset Color;

echo;

#

#

#

#

# == Notes ==

#

# [Note 1]

# [

# For explanation on the color code:

# |\*| Coloring test utility: https://unix.stackexchange.com/a/643536 ]

#

cat "$0" 1>&2;

#

# = 256-color test (old) =

#

# [

# |\*| Source: https://unix.stackexchange.com/a/643715

# |\*| Source (original): https://misc.flogisoft.com/bash/tip\_colors\_and\_formatting#colors2

# |\*| Last update: CE 2021-05-15 03:17 UTC ]

#

#

# Basically a replicate of the original with no logic change. Left there mostly for reference.

#

#

#

#

# == Implementation ==

#

# Colors (0..255):

Colors="$(

i=0;

while

echo "$i";

[ $i -lt 255 ];

do

i=$(( $i + 1 ));

done;

)";

echo;

for x0 in \

'38' '48'; # Foreground / Background

do {

for Color in \

$Colors;

do {

printf '\e['"$x0"';5;%sm %3s ' \

"$Color" "$Color";

# 6 entries per line:

[ $(( ($Color + 1) % 6 )) -eq 4 ] && echo -e '\e[m';

};

done;

echo;

};

done;

unset Colors x0 Color;