

Результаты опроса разработчиков 2024 года уже опубликованы! Посмотреть результаты

## Как изменить цвет вывода echo в Linux

Спросил 13 лет, 2 месяца назад Изменено 1 год, 3 месяца назад Просмотрено 2,0 млн раз



Я пытаюсь напечатать текст в терминале с помощью команды echo.

**2784**

Я хочу напечатать текст красным цветом. Как это сделать?



[линукс](#) [Баш](#) [командная строка](#) [эхо](#) [терминальный цвет](#)



Делиться Улучшить этот вопрос отредактировано 13 дек. 2017 г. в 4:12 задано 10 мая 2011 г. в 9:07

Следовать



whackamadoodle3000

6,738 4 28 45



Сатиш.дроид

31,1 тыс. 10 35 34

**64** Эта ссылка очень полезна: [misc.flogisoft.com/bash/tip\\_colors\\_and\\_formatting](http://misc.flogisoft.com/bash/tip_colors_and_formatting).  
– [Питикос](#) 22 окт. 2014 г. в 10:32

**2** echo -e "plain \e[0;31mКРАСНОЕ СООБЩЕНИЕ \e[0m сброс" – [Кончог](#) 13 сен 2022 г. в 17:25

Отсортировано по:

33 ответа

Наивысший балл (по умолчанию)



1

2

Следующий



Вы можете использовать следующие [управляющие коды ANSI](#):



**3711**

Black	0;30	Dark Gray	1;30
Red	0;31	Light Red	1;31
Green	0;32	Light Green	1;32
Brown/Orange	0;33	Yellow	1;33
Blue	0;34	Light Blue	1;34
Purple	0;35	Light Purple	1;35
Cyan	0;36	Light Cyan	1;36
Light Gray	0;37	White	1;37



А затем используйте их в своем сценарии следующим образом:

```
# ----- constant part!
# vvvv vvvv-- the code from above
RED='\033[0;31m'
NC='\033[0m' # No Color
printf "I ${RED}love${NC} Stack Overflow\n"
```

который печатает love красным цветом.

Из комментария @james-lim следует, что **если вы используете echo команду, обязательно используйте -e флаг, разрешающий экранирование с помощью обратной косой черты .**

```
#      ----- constant part!
#      vvvv vvvv-- the code from above
RED='\033[0;31m'
NC='\033[0m' # No Color
echo -e "I ${RED}love${NC} Stack Overflow"
```

**Примечание :** не добавляйте "\n" при использовании echo , если только вы не хотите добавить дополнительную пустую строку.

Делиться Улучшить этот ответ  
Следовать

отредактировано 12 апр. 2023 г.  
в 20:33

ответил 10 мая 2011 г. в 9:11  
 Тобиас  
37,7 тыс. 1 21 23

 BluePigeon  
1,802 3 17 40

- 29 У меня не работает – вывод: \e[0;31mHello Stackoverflow\e[0m – [Бен Гарольд](#) 9 мая 2013 в 22:01
- 236 Вы пробовали с "-e"? Он говорит echo включить экранирование обратной косой черты.  
– [Джеймс Лим](#) 14 мая 2013 в 13:10
- 182 В MacOSX использование \x1B вместо \e . \033 будет приемлемым для всех платформ.  
– [Сяо](#) 19 июня 2013 г. в 6:04
- 6 В файле свойств ant используйте upicode для esacpe, например red=\u001b[0;31m  
– [shonky](#) пользователь linux 1 окт. 2013 г. в 0:55
- 43 Как и msanford, созданный для tput, вот "ANSI-Rainbow" for (( i = 30; i < 38; i++ )); do echo -e "\033[0;\${i}m Normal: (\$i); \033[1;\${i}m Light: (\$i)"; done  
– [каждый человек](#) 28 января 2016 г. в 21:28

### некоторые переменные, которые вы можете использовать:

1736

```
# Reset
Color_Off='\033[0m'          # Text Reset

# Regular Colors
Black='\033[0;30m'           # Black
Red='\033[0;31m'              # Red
Green='\033[0;32m'            # Green
Yellow='\033[0;33m'            # Yellow
Blue='\033[0;34m'             # Blue
Purple='\033[0;35m'            # Purple
Cyan='\033[0;36m'              # Cyan
White='\033[0;37m'             # White

# Bold
BBlack='\033[1;30m'            # Black
BRed='\033[1;31m'              # Red
BGreen='\033[1;32m'             # Green
```

```

BYellow='\033[1;33m'          # Yellow
BBlue='\033[1;34m'           # Blue
BPurple='\033[1;35m'          # Purple
BCyan='\033[1;36m'           # Cyan
BWhite='\033[1;37m'          # White

# Underline
UBlack='\033[4;30m'          # Black
URed='\033[4;31m'            # Red
UGreen='\033[4;32m'           # Green
UYellow='\033[4;33m'          # Yellow
UBlue='\033[4;34m'            # Blue
UPurple='\033[4;35m'          # Purple
UCyan='\033[4;36m'            # Cyan
UWhite='\033[4;37m'           # White

# Background
On_Black='\033[40m'          # Black
On_Red='\033[41m'             # Red
On_Green='\033[42m'            # Green
On_Yellow='\033[43m'           # Yellow
On_Blue='\033[44m'             # Blue
On_Purple='\033[45m'            # Purple
On_Cyan='\033[46m'              # Cyan
On_White='\033[47m'             # White

# High Intensity
IBlack='\033[0;90m'          # Black
IRed='\033[0;91m'              # Red
IGreen='\033[0;92m'             # Green
IYellow='\033[0;93m'            # Yellow
IBlue='\033[0;94m'              # Blue
IPurple='\033[0;95m'             # Purple
ICyan='\033[0;96m'              # Cyan
IWhite='\033[0;97m'             # White

# Bold High Intensity
BIBlack='\033[1;90m'          # Black
BIRed='\033[1;91m'              # Red
BIGreen='\033[1;92m'             # Green
BIYellow='\033[1;93m'            # Yellow
BIBlue='\033[1;94m'              # Blue
BIPurple='\033[1;95m'             # Purple
BICyan='\033[1;96m'              # Cyan
BIWhite='\033[1;97m'             # White

# High Intensity backgrounds
On_IBlack='\033[0;100m'         # Black
On_IRed='\033[0;101m'             # Red
On_IGreen='\033[0;102m'            # Green
On_IYellow='\033[0;103m'           # Yellow
On_IBlue='\033[0;104m'              # Blue
On_IPurple='\033[0;105m'             # Purple
On_ICyan='\033[0;106m'              # Cyan
On_IWhite='\033[0;107m'             # White

```



**Экранированный символ в bash , hex и octal соответственно:**

	bash	hex	octal	NOTE	
start	\e	\x1b	\033		

start	\E	\x1B	-	x cannot be capital	
end	\e[0m	\x1b[0m	\033[0m		
end	\e[m	\x1b[m	\033[m	0 is appended if you omit it	

### краткий пример:

color	bash	hex	octal	NOTE
start green	\e[32m<text>   \x1b[32m<text>   \033[32m<text>			m is NOT optional
reset	<text>\e[0m   <text>\x1b[0m   <text>\033[0m			o is optional (do it as best practice)

### исключение bash:

Если вы собираетесь использовать эти коды в своих специальных переменных *bash*

- PS0
- PS1
- PS2 (= это для подсказки)
- PS4

вам следует добавить дополнительные экранированные символы, чтобы *Баш* может интерпретировать их правильно. Без этого добавления дополнительных экранирующих символов это работает, но вы столкнетесь с проблемами при использовании Ctrl + g для поиска в своей истории.

### правило исключения для bash

Вы должны добавить \[ перед любым начальным кодом ANSI и добавить \] после любого конечного.

Пример:

при обычном использовании: \033[32mThis is in green\033[0m

для PS0/1/2/4: \[\033[32m\]This is in green\[\033[m\]

\[ служит для начала последовательности **непечатаемых** символов  
 \] служит для конца последовательности **непечатаемых** символов

Совет: чтобы запомнить, вы можете сначала добавить \[\], а затем вставить между ними свой код ANSI:

- \[start-ANSI-code\]
- \[end-ANSI-code\]

## тип цветовой последовательности:

1. 3/4 бита
2. 8 бит
3. 24 бита

Прежде чем погрузиться в эти цвета, вам следует знать о 4 режимах с этими кодами:

### 1. цветовой режим

Он изменяет стиль цвета, а не текста. Например, сделать цвет ярче или темнее.

- 0; перезагрузить
- 1; легче обычного
- 2; темнее обычного

Этот режим не поддерживается широко. Он полностью поддерживается в Gnome-Terminal.

### 2. текстовый режим

Этот режим предназначен для изменения стиля текста, а НЕ цвета.

- 3; курсив
- 4; подчеркивание
- 5; мигает (медленно)
- 6; мигает (быстро)
- 7; обеспечить регресс
- 8; скрывать
- 9; вычеркнуть

и почти поддерживаются.

Например, KDE-Konsole поддерживает 5;, а Gnome-Terminal нет, а Gnome поддерживает 8;, а KDE нет.

### 3. режим переднего плана

Этот режим предназначен для раскрашивания переднего плана.

### 4. фоновый режим

Этот режим предназначен для раскрашивания фона.



В таблице ниже приведены сводные данные по 3/4-битной версии ANSI-цвета.

color-mode	octal	hex	bash	description	example (= in
octal)	NOTE				
0   \033[0m   \x1b[0m   \e[0m   reset any affect   echo -e "\033[0m"					
0m equals to m					
1   \033[1m     light (= bright)   echo -e					
"\033[1m####\033[m"   -					
2   \033[2m     dark (= fade)   echo -e					
"\033[2m####\033[m"   -					
text-mode   ~       ~   ~					
~					
3   \033[3m     italic   echo -e					
"\033[3m####\033[m"					
4   \033[4m     underline   echo -e					
"\033[4m####\033[m"					
5   \033[5m     blink (slow)   echo -e					
"\033[5m####\033[m"					
6   \033[6m     blink (fast)   ?					
not wildly support					
7   \003[7m     reverse   echo -e					
"\003[7m####\033[m"   it affects the background/foreground					
8   \033[8m     hide   echo -e					
"\033[8m####\033[m"   it affects the background/foreground					
9   \033[9m     cross   echo -e					
"\033[9m####\033[m"					
foreground   ~       ~   ~					
~					
30   \033[30m     black   echo -e					
"\033[30m####\033[m"					
31   \033[31m     red   echo -e					
"\033[31m####\033[m"					
32   \033[32m     green   echo -e					
"\033[32m####\033[m"					
33   \033[33m     yellow   echo -e					
"\033[33m####\033[m"					
34   \033[34m     blue   echo -e					
"\033[34m####\033[m"					
35   \033[35m     purple   echo -e					
"\033[35m####\033[m"   real name: magenta = reddish-purple					
36   \033[36m     cyan   echo -e					
"\033[36m####\033[m"					
37   \033[37m     white   echo -e					
"\033[37m####\033[m"					
24-bit	38   8/24   This is for special use of 8-bit or				
background   ~       ~   ~					
~					

40	\033[40m		black		echo -e
"\033[40m#####\033[m"					
41	\033[41m		red		echo -e
"\033[41m#####\033[m"					
42	\033[42m		green		echo -e
"\033[42m#####\033[m"					
43	\033[43m		yellow		echo -e
"\033[43m#####\033[m"					
44	\033[44m		blue		echo -e
"\033[44m#####\033[m"					
45	\033[45m		purple		echo -e
"\033[45m#####\033[m"		real name: magenta	= reddish-purple		
46	\033[46m		cyan		echo -e
"\033[46m#####\033[m"					
47	\033[47m		white		echo -e
"\033[47m#####\033[m"					
48	8/24			This is for special use of 8-bit or 24-bit	

В таблице ниже приведены сводные данные по **8-битной** версии ANSI-цвета.

foreground	octal	hex	bash	description	example
NOTE					
0-7	\033[38;5	\x1b[38;5	\e[38;5	standard. normal	echo -e
"\033[38;5;1m#####\033[m"					
8-15				standard. light	echo -e
"\033[38;5;9m#####\033[m"					
16-231				more resolution	echo -e
"\033[38;5;45m#####\033[m"		has no specific pattern			
232-255					echo -e
"\033[38;5;242m#####\033[m"		from black to white			
NOTE					
foreground	octal	hex	bash	description	example
0-7				standard. normal	echo -e
"\033[48;5;1m#####\033[m"					
8-15				standard. light	echo -e
"\033[48;5;9m#####\033[m"					
16-231				more resolution	echo -e
"\033[48;5;45m#####\033[m"					
232-255					echo -e
"\033[48;5;242m#####\033[m"		from black to white			

### 8-битный быстрый тест:

```
for code in {0..255}; do echo -e "\e[38;05;${code}m $code: Test"; done
```

В таблице ниже приведены сводные данные по **24-битной** версии ANSI-color.

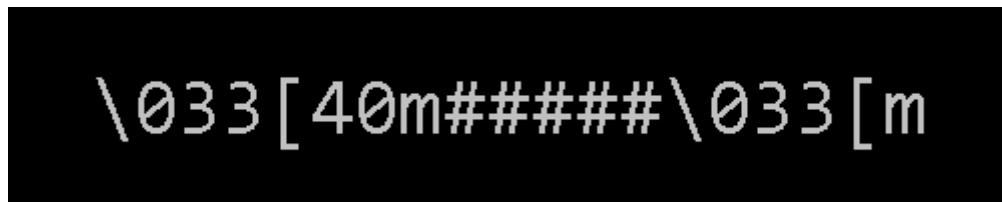
foreground	octal	hex	bash	description	example
NOTE					
0-255	\033[38;2   \x1b[38;2   \e[38;2   R = red   echo -e				
'\033[38;2;255;0;02m####\033[m'			R=255, G=0, B=0		
0-255	\033[38;2   \x1b[38;2   \e[38;2   G = green   echo -e				
'\033[38;2;0;255;02m####\033[m'			R=0, G=255, B=0		
0-255	\033[38;2   \x1b[38;2   \e[38;2   B = blue   echo -e				
'\033[38;2;0;0;2552m####\033[m'			R=0, G=0, B=255		
background	octal	hex	bash	description	example
NOTE					
0-255	\033[48;2   \x1b[48;2   \e[48;2   R = red   echo -e				
'\033[48;2;255;0;02m####\033[m'			R=255, G=0, B=0		
0-255	\033[48;2   \x1b[48;2   \e[48;2   G = green   echo -e				
'\033[48;2;0;255;02m####\033[m'			R=0, G=255, B=0		
0-255	\033[48;2   \x1b[48;2   \e[48;2   B = blue   echo -e				
'\033[48;2;0;0;2552m####\033[m'			R=0, G=0, B=255		

## Некоторые скриншоты

передний план 8-битного резюме в .gif



фон 8-битный резюме в .gif



## Цветовая сводка с их значениями

## Colored text (ANSI)

Color	Foreground	Background
Black		30
Red		31
Green		32
Yellow		33
Blue		34
Magenta		35
Cyan		36
White		37

## Styled text (ANSI)

Style	Value
No Style	0
Bold	1
Low Intensity	2
Underline	4
Blinking	5
Reverse	7
Invisible	8

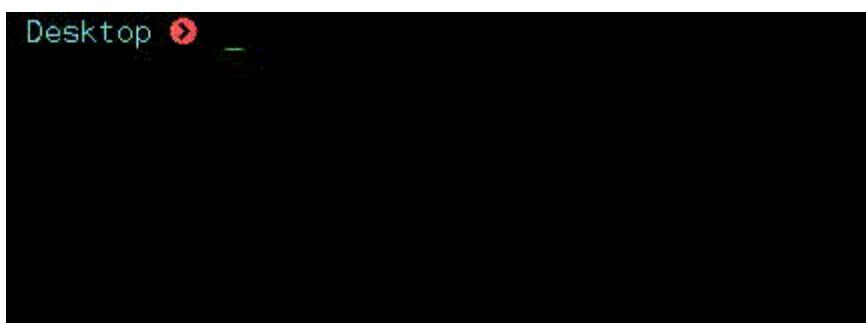
## Styled text (tput)

Color	setaf	setab
Black	0	0
Red	1	1
Green	2	2
Yellow	3	3
Blue	4	4
Magenta	5	5
Cyan	6	6
White	7	7

## Styled text (tput)

Style	Command
Foreground	tput setaf [0-7]
Background	tput setab [0-7]
No Style	tput sgr0
Bold	tput bold
Low Intensity	tput dim
Underline	tput smul
Blinking	tput blink
Reverse	tput rev

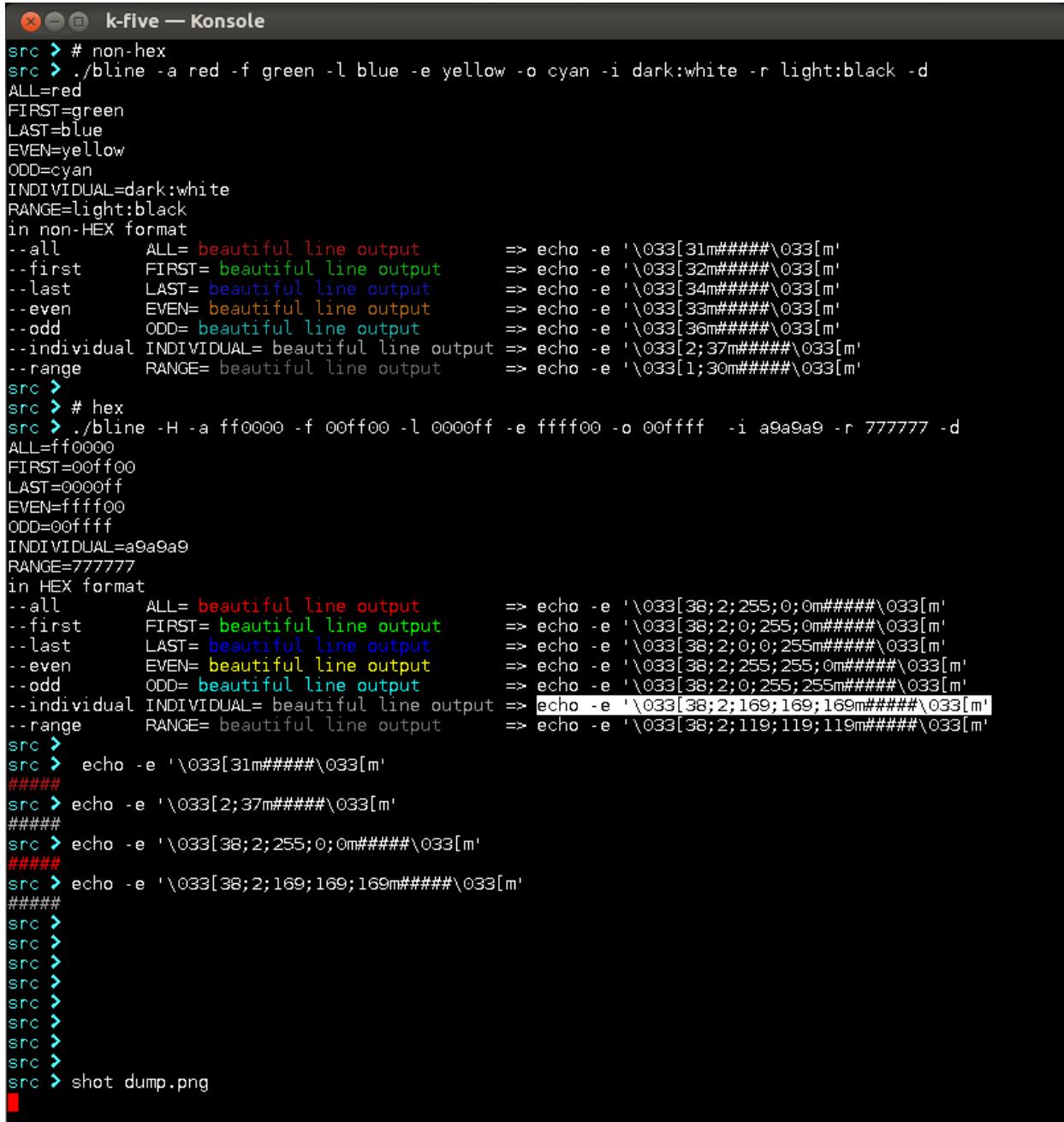
blinking on KDE-терминал



echo С простой код 'С', который показывает вам больше

```
ideas ./cecho
Usage: [color-name] [some-text] [endl]
ideas
ideas ./cecho green *.txt
ansi_color.txt empty.txt file.txt ListOut3.txt List.txt myData.txt
ideas ./cecho green *.txt endl
ansi_color.txt
empty.txt
file.txt
ListOut3.txt
List.txt
myData.txt
ideas ./cecho green *.txt endl | cat -n
1 ansi_color.txt
2 empty.txt
3 file.txt
4 ListOut3.txt
5 List.txt
6 myData.txt
ideas ./cecho random *.txt endl | cat -n
1 ansi_color.txt
2 empty.txt
3 file.txt
4 ListOut3.txt
5 List.txt
6 myData.txt
ideas ./cecho random *.txt endl | cat -n
1 ansi_color.txt
2 empty.txt
3 file.txt
4 ListOut3.txt
5 List.txt
6 myData.txt
ideas ./cecho random *.txt endl | cat -n
1 ansi_color.txt
2 empty.txt
3 file.txt
4 ListOut3.txt
5 List.txt
6 myData.txt
ideas _
```

**bline** С более продвинутый инструмент, который я разработал для работы с этими цветами:



```

src > # non-hex
src > ./bline -a red -f green -l blue -e yellow -o cyan -i dark:white -r light:black -d
ALL=red
FIRST=green
LAST=blue
EVEN=yellow
ODD=cyan
INDIVIDUAL=dark:white
RANGE=light:black
in non-HEX format
--all      ALL= beautiful line output      => echo -e '\033[31m#####\033[m'
--first    FIRST= beautiful line output    => echo -e '\033[32m#####\033[m'
--last     LAST= beautiful line output    => echo -e '\033[34m#####\033[m'
--even     EVEN= beautiful line output    => echo -e '\033[33m#####\033[m'
--odd      ODD= beautiful line output    => echo -e '\033[36m#####\033[m'
--individual INDIVIDUAL= beautiful line output => echo -e '\033[2;37m#####\033[m'
--range    RANGE= beautiful line output    => echo -e '\033[1;30m#####\033[m'
src >
src > # hex
src > ./bline -H -a ff0000 -f coffeeo -l 000off -e fffffoo -o 00ffff -i a9a9a9 -r 777777 -d
ALL=ff0000
FIRST=00ff00
LAST=0000ff
EVEN=ffff00
ODD=00ffff
INDIVIDUAL=a9a9a9
RANGE=777777
in HEX format
--all      ALL= beautiful line output      => echo -e '\033[38;2;255;0;0m#####\033[m'
--first    FIRST= beautiful line output    => echo -e '\033[38;2;0;255;0m#####\033[m'
--last     LAST= beautiful line output    => echo -e '\033[38;2;0;0;255m#####\033[m'
--even     EVEN= beautiful line output    => echo -e '\033[38;2;255;255;0m#####\033[m'
--odd      ODD= beautiful line output    => echo -e '\033[38;2;0;255;255m#####\033[m'
--individual INDIVIDUAL= beautiful line output => echo -e '\033[38;2;169;169;169m#####\033[m'
--range    RANGE= beautiful line output    => echo -e '\033[38;2;119;119;119m#####\033[m'
src >
src > echo -e '\033[31m#####\033[m'
#####
src > echo -e '\033[2;37m#####\033[m'
#####
src > echo -e '\033[38;2;255;0;0m#####\033[m'
#####
src > echo -e '\033[38;2;169;169;169m#####\033[m'
#####
src >
src > shot dump.png
■

```

## СНИМОК В ЦВЕТОВОМ РЕЖИМЕ

```

~ ➜ echo -e 'How \033[2;32mare you\033[m today?'; # fade foreground
How are you today?
~ ➜ echo -e 'How \033[0;32mare you\033[m today?'; # normal foreground
How are you today?
~ ➜ echo -e 'How \033[1;32mare you\033[m today?'; # bright foreground
How are you today?

```

## ТЕКСТОВЫЙ РЕЖИМ СНИМОК

```

~ ⚡ # only text mode
~ ⚡ echo -e 'How \033[3mage you\033[m today?'; # italic
How are you today?
~ ⚡ echo -e 'How \033[4mage you\033[m today?'; # underline
How are you today?
~ ⚡ echo -e 'How \033[5mage you\033[m today?'; # blinking
How are you today?
~ ⚡ # blinking works on KDE konsole but not here
~ ⚡ # but cross-word works here but not on KDE
~ ⚡ echo -e 'How \033[9mage you\033[m today?'; # [9 crossing
How areyou today?

```

## объединение в порядке

```

~ ⚡ echo -e 'How \033[9;32mage you\033[m today?'; # set foreground
How areyou today?
~ ⚡ echo -e 'How \033[9;42mage you\033[m today?'; # set background
How areyou today?
~ ⚡ echo -e 'How \033[9;31;42mage you\033[m today?'; # set both red:green
How areyou today?

```

## [больше снимков](#)

**Советы и рекомендации для продвинутых пользователей и программистов:**

**Можем ли мы использовать эти коды в языке программирования?**

Да, можно. Я испытал в `Баш`, `с`, `c++`, `г` `perl`, `питон`

**Замедляют ли они скорость работы программы?**

Я думаю нет.

**Можно ли использовать их в Windows?**

3/4-бит Да, если вы скомпилируете код с `gcc`

[некоторыми снимками экрана на Win-7](#)

**Как рассчитать длину кода?**

`\033[ = 2, остальные части 1`

**Где можно использовать эти коды?**

Везде, где есть `tty` интерпретатор

`xterm`, `gnome-terminal`, `kde-terminal`, `mysql-client-CLI` и т. д.

Например, если вы хотите раскрасить вывод с помощью `mysql`, вы можете использовать `Perl`

```

#!/usr/bin/perl -n
print "\033[1m\033[31m\$1\033[36m\$2\033[32m\$3\033[33m\$4\033[m" while /([|+-]+)|([0-

```

```
9]+)|([a-zA-Z_]+)|([^\w])/g;
```

сохраните этот код в файле с именем: pcc (= Perl Colorize Character), а затем поместите файл в valid PATH и используйте его где угодно.

```
ls | pcc
df | pcc
```

внутри mysql сначала зарегистрируйте его pager, а затем попробуйте:

```
[user2:db2] pager pcc
PAGER set to 'pcc'
[user2:db2] select * from table-name;
```

```
[user2:db2]
[user2:db2] pager pcc
PAGER set to 'pcc'
[user2:db2] select * from w3;
+----+-----+
| id | name |
+----+-----+
| 2  | second |
| 3  | third  |
| 4  | fourth |
| 5  | fifth  |
| 6  | last-one |
+----+-----+
5 rows in set (0.00 sec)
```

Он **НЕ** обрабатывает Unicode.

### Эти коды выполняют только раскрашивание?

Нет, они могут делать много интересных вещей. Попробуйте:

```
echo -e '\033[2K' # clear the screen and do not move the position
```

или:

```
echo -e '\033[2J\033[u' # clear the screen and reset the position
```

Множество новичков хотят очистить экран, system( "clear" ) поэтому вы можете использовать это вместо system(3) вызова

### Доступны ли они в Unicode?

Да. \u001b

### Какой вариант этих цветов предпочтительнее?

Он прост в использовании 3/4-bit, но гораздо точнее и красивее в использовании 24-bit .

Если у вас нет опыта работы с `html` Итак, вот краткий урок:

24 бита означают: 00000000 и 00000000 и 00000000 . Каждые 8 бит предназначены для определенного цвета.

1..8 предназначены для  и 9..16 для  и 17..24 для 

Так что в `html` #FF0000 означает  и вот оно: 255;0;0

в `html` #00FF00 означает , что здесь: 0;255;0

имеет ли это смысл? какой цвет вы хотите объединить с этими тремя 8-битными значениями.

ссылка:

[Wikipedia](#)

[ANSI escape-последовательности](#)

[tldp.org](#)

[tldp.org](#)

[misc.flogisoft.com](#)

некоторые блоги/веб-страницы, которые я не помню

Делиться Улучшить этот ответ

отредактировано 26 сен 2021 в

4:04

ответил 9 марта 2015 г. в 9:02



Шакиба Мошири

23 тыс. 3 35 48

3 @NeilGuyLindberg **нет восьмеричных литералов**, эта ошибка является частью Node.js, а не самого eslist. Вы можете использовать x1B[ ее для устранения. – [Шакиба Мошири](#) 3 окт. 2019 г. в 6:45

1 @ShakibaMoshiri Из ответа не ясно, как комбинировать цвета, пока вы внимательно не прочитаете [escape-последовательности ANSI](#) . Просто для справки: echo -e "\033[97;44;1m text \033[m" выведет жирный (;1) белый текст (;97) на синем фоне (;44) и \033[0m сбросит все атрибуты текста (0). Также это зависит от цветовой схемы терминала. – [Йотер](#) 23 апр. 2020 г. в 12:39 

Почему в этом коротком примере восьмеричного столбца <text>\033[0m вместо буквы «о» стоит <text>\033[0m буква «о», а не цифра «ноль»? – [полночь](#) 2 дек. 2023 г. в 14:09

@midnite это **опечатка** вы можете обновить/отредактировать ответ – [Шакиба Мошири](#) 4 дек. 2023 г. в 20:06



Вы можете использовать замечательную `tput` команду (предложенную в [ответе Игнасио](#)) для создания кодов управления терминалом для самых разных вещей.

1321



## Применение



Конкретные `tput` подкоманды обсуждаются далее.



## Прямой

Вызов трут как часть последовательности команд:

```
tput setaf 1; echo "this is red text"
```

Используйте ; вместо &&, чтобы в случае трут ошибок текст все равно отображался.

## Переменные оболочки

Другой вариант – использовать переменные оболочки:

```
red=`tput setaf 1`  
green=`tput setaf 2`  
reset=`tput sgr0`  
echo "${red}red text ${green}green text${reset}"
```

трут создает последовательности символов, которые интерпретируются терминалом как имеющие особое значение. Они не будут показаны сами по себе. Обратите внимание, что их все еще можно сохранять в файлы или обрабатывать как ввод другими программами, а не терминалом.

## Замена команды

Возможно, будет удобнее вставить трут вывод непосредственно в echo строки, используя [подстановку команд](#) :

```
echo "$(tput setaf 1)Red text $(tput setab 7)and white background$(tput sgr 0)"
```

## Пример

Приведенная выше команда выводит следующее в Ubuntu:

```
$ echo "$(tput setaf 1)Red text $(tput setab 7)and white background$(tput sgr 0)"  
Red text and white background  
$
```

## Команды цвета переднего плана и фона

```
tput setab [1-7] # Set the background colour using ANSI escape  
tput setaf [1-7] # Set the foreground colour using ANSI escape
```

Цвета следующие:

Num	Colour	#define	R G B
0	black	COLOR_BLACK	0,0,0
1	red	COLOR_RED	1,0,0

```

2   green      COLOR_GREEN      0,1,0
3   yellow     COLOR_YELLOW    1,1,0
4   blue       COLOR_BLUE      0,0,1
5   magenta    COLOR_MAGENTA   1,0,1
6   cyan       COLOR_CYAN     0,1,1
7   white      COLOR_WHITE    1,1,1

```

Существуют также не-ANSI версии функций настройки цвета ( `setb` вместо `setab` , и `setf` вместо `setaf` ), которые используют другие числа, здесь не приведенные.

## Команды текстового режима

```

tput bold      # Select bold mode
tput dim       # Select dim (half-bright) mode
tput smul      # Enable underline mode
tput rmul      # Disable underline mode
tput rev       # Turn on reverse video mode
tput smso      # Enter standout (bold) mode
tput rmsq      # Exit standout mode

```

## Команды перемещения курсора

```

tput cup Y X # Move cursor to screen position X,Y (top left is 0,0)
tput cuf N   # Move N characters forward (right)
tput cub N   # Move N characters back (left)
tput cuu N   # Move N lines up
tput ll      # Move to last line, first column (if no cup)
tput sc      # Save the cursor position
tput rc      # Restore the cursor position
tput lines   # Output the number of lines of the terminal
tput cols   # Output the number of columns of the terminal

```

## Очистить и вставить команды

```

tput ech N   # Erase N characters
tput clear   # Clear screen and move the cursor to 0,0
tput el 1    # Clear to beginning of line
tput el      # Clear to end of line
tput ed      # Clear to end of screen
tput ich N   # Insert N characters (moves rest of line forward!)
tput il N    # Insert N lines

```

## Другие команды

```

tput sgr0    # Reset text format to the terminal's default
tput bel     # Play a bell

```

В [Compiz Wobbly Windows](#) эта `bel` команда заставляет терминал на секунду дрожать, чтобы привлечь внимание пользователя.

## Скрипты

`tput` принимает скрипты, содержащие по одной команде в строке, которые выполняются по порядку перед `tput` выходом.

Избегайте создания временных файлов, выводя многострочную строку и передавая ее по конвейеру:

```
echo -e "setf 7\nsetb 1" | tput -S # set fg white and bg red
```

## Смотрите также

- Видеть [man 1 tput](#)
- [man 5 terminfo](#) Полный список команд и более подробную информацию об этих параметрах см. здесь . (Соответствующая `tput` команда указана в Сар-наме столбце огромной таблицы, которая начинается со строки 81.)

Делиться Улучшить этот ответ  
Следовать

отредактировано 28 июня 2018 г.  
в 14:52

ответил 7 января 2014 г. в  
22:25

 тканевый человек  
19,1 тыс. 20 66 86

 Дрю Ноукс  
308 тыс. 167 690  
758

- 
- 25 Отличный ответ. Это тот, который помог мне больше всего. Для тех, кто еще задавался вопросом о том, что задавался я, это `$()` подстановка [команды](#) . Все, `tput af 1` что делает, это генерирует строку цветового кода, но коды не являются печатаемыми символами, поэтому ввод `tput af 1` только приведет к пустой строке вывода.  
– [Крис Миддлтон](#) 25 июл 2014 в 15:45 
- 6 Примечание: если вы используете CygWin и у вас не установлен `tput ncurses` – [Энрико](#) 12 дек. 2014 г. в 18:56
- 5 `tput` также работает внутри `sed` для разбора хлама в разборчивый, красочный хлам: [gist.github.com/nickboldt/fab71da10bd5169ffdfa](https://gist.github.com/nickboldt/fab71da10bd5169ffdfa) – [Никболдт](#) 5 февр. 2015 г. в 19:06
- 4 Полный список цветов `tput` можно [найти в этом ответе на Unix StackExchange.](#) – [Андрей](#) 21 января 2017 г. в 23:10 
- 2 Отличный ответ, но стоит подчеркнуть, насколько лучше хранить выходные данные `tput` в переменных оболочки, как вы показываете, чем встраивать `tput`, как вы показываете сразу после этого. Каждый интерполированный вызов `tput` порождает процесс, и для любого серьезного использования цветов для выделения и т. д. для многих строк вызов `tput` несколько раз для каждой строки создает *серьезное видимое замедление*. Не говоря уже о том, что встраивание `tput` гораздо менее читабельно. – [пользователь1169420](#) 17 авг. 2022 в 16:45

Используйте `tput` с `setaf` возможностью и параметром 1 .

240

```
echo "$(tput setaf 1)Hello, world$(tput sgr0)"
```



Делиться Улучшить этот ответ Следовать

ответил 10 мая 2011 г. в 9:10



Игнасио Васкес-Абрамс  
792 тыс. 157 1.4k  
1.4k

13 Это должен быть лучший вариант. *tput* считывает информацию терминала и отображает для вас правильно экранированный код ANSI. Такой код \033[31m нарушит работу библиотеки *readline* в некоторых терминалах. – Тянь Чэнь 6 мар. 2013 г. в 7:39

54 Исследуйте цвета с помощью простого цикла (увеличьте і верхнюю границу для получения большего количества оттенков): for (( i = 0; i < 17; i++ )); do echo "\$(tput setaf \$i)This is (\$i) \$(tput sgr0)"; done – мсэнфорд 22 января 2014 г. в 15:41

4 Вот HOWTO по кодам *tput*: [tldp.org/HOWTO/Bash-Prompt-HOWTO/x405.html](https://tldp.org/HOWTO/Bash-Prompt-HOWTO/x405.html) – Максивб 4 авг. 2014 г., 18:38

Согласен с @TianChen, такой код \033[31m также будет выдавать некоторые нерелевантные символы, когда программа, используемая для вывода текста, несовместима с такими командами. С другой стороны, *tput* + *setaf* команды не будут, оставляя вывод полностью читаемым. Смотрите комментарий @BenHarold, в котором говорится: "У меня не работает -- вывод: \e[0;31mHello Stackoverflow\e[0m" – Эль Тиди 5 окт. 2020 г. в 10:42

Хм, в чем разница? Оба варианта создают одинаковые начальные команды изменения цвета, но отличаются завершающей командой, которая сбрасывает. Прямые escape-последовательности создают именно тот двоичный вывод, который вы ожидаете, учитывая то, что было введено – \ESC[m . *tput* Альтернатива на самом деле создает немного другую (и более длинную) последовательность \ESC(B\ESC[m . Сравните разницу между тем, что опубликовано здесь, и ее \033[31mHello, world\033[m . *sgr0* на самом деле создает две завершающие escape-последовательности. Почему это предпочтительнее первого варианта, когда дело касается проблем с ее *readline*? – ШерреллБК 14 июля 2022 г., 2:24



`echo -e "\033[31m Hello World"`

200

Управляет [31m цветом текста:



- 30 - 37 устанавливает цвет *переднего плана*
- 40 - 47 устанавливает цвет *фона*



Более полный список цветовых кодов [можно найти здесь](#).

Хорошей практикой является восстановление цвета текста до значения, указанного \033[0m в конце строки.

Делиться Улучшить этот ответ Следовать

отредактировано 19 окт. 2017 г. в 15:53 ответил 10 мая 2011 г. в 9:10



Фрэнк Н.

10,2 тыс. 4 87 115



Heo

5,383 2 24 29

4 echo -e "\033[31m Hello World", [31m - это цвет - [Heo](#) 10 мая 2011 г., 9:13

спасибо, что добавили кое-что, что я могу скопировать и вставить - [TheGaldozer](#) 14 июл. 2022 в 16:49

Мне нравится ваш ответ, потому что он лаконичен :) - [Посмотрим](#) 13 марта 2023 г. в 10:43



Я просто объединил удачные решения во всех решениях и в итоге получил:

114

```
cecho(){
    RED="\033[0;31m"
    GREEN="\033[0;32m" # <-- [0 means not bold
    YELLOW="\033[1;33m" # <-- [1 means bold
    CYAN="\033[1;36m"
    # ... Add more colors if you like

    NC="\033[0m" # No Color

    # printf "${(P)1}${(2) ${NC}}\n" # <-- zsh
    printf "${!1}${(2) ${NC}}\n" # <-- bash
}
```



И вы можете просто назвать это так:

```
cecho "RED" "Helloworld"
```

**Helloworld**

Делиться Улучшить этот ответ

отредактировано 8 июня 2022 г.  
в 18:19

ответил 24 нояб. 2018 г. в  
23:09



floer32

2,250 4 30 52



ndgrwnaguib

5,965 3 29 52

2 Очень практично, мне просто пришлось заменить одинарные кавычки на двойные для GREEN, YELLOW, NC, чтобы это заработало в моем скрипте. - [ionescu77](#) 21 марта 2019 г. в 12:42

2 У меня возникло несколько проблем при отображении содержимого файла. Замена на printf помогла echo мне решить эту проблему. - [Ребенок](#) 26 ноября 2021 г., 22:30

1 в zsh я получаю: Плохая замена - [Адриан Лопес](#) 26 дек. 2021 г. в 21:54

1 @AdrianLopez спасибо, что заметили это! [косвенная ссылка на переменную](#) в ZSH используется \${(P)1} вместо \${!1} for bash. Я отредактировал ответ и включил оба. - [ndgrwnaguib](#) 20 января 2022 г. в 18:12

1 Спасибо. Полезно знать о [косвенной ссылке на переменную](#) : \${!1} как отмечено выше! В противном случае это все стандартные функции bash. - [Нишант](#) 27 июля 2022 в 10:00 



Мой комментарий к ответу Тобиаса:

57

```
# Color
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[0;33m'
NC='\033[0m' # No Color

function red {
    printf "${RED}@$${NC}\n"
}

function green {
    printf "${GREEN}@$${NC}\n"
}

function yellow {
    printf "${YELLOW}@$${NC}\n"
}
```

```
$ echo $(red apple) $(yellow banana) $(green kiwi)
apple banana kiwi
```

Делиться Улучшить этот ответ Следовать

ответил 20 января 2021 г. в  
17:56



белый

2,940 1 23 39



Это **переключение цвета** \033[ . Смотреть [историю](#) .

43

Цветовые [коды](#) такие: 1;32 (светло-зеленый), 0;34 (синий), 1;34 (светло-голубой) и т. д.



Мы завершаем последовательности цветов переключателем цвета \033[ и 0m , кодом без цвета. Так же, как открытие и закрытие вкладок в языке разметки.

```
SWITCH="\033["
NORMAL="${SWITCH}0m"
YELLOW="${SWITCH}1;33m"
echo "${YELLOW}hello, yellow${NORMAL}"
```

Простое echo решение цветовой функции:

```
cecho() {
    local code="\033["
    case "$1" in
        black | bk) color="${code}0;30m";;
        red | r) color="${code}1;31m";;
        green | g) color="${code}1;32m";;
        yellow | y) color="${code}1;33m";;
        blue | b) color="${code}1;34m";;
        purple | p) color="${code}1;35m";;
```

```

cyan    |   c) color="${code}1;36m";;
gray    | gr) color="${code}0;37m";;
*) local text="$1"
esac
[ -z "$text" ] && local text="$color$2${code}0m"
echo "$text"
}

cecho "Normal"
cecho y "Yellow!"

```

Делиться Улучшить этот ответ

Следовать

отредактировано 23 мая 2017 г.

в 12:18



Сообщество Бот

1 1

ответил 25 февр. 2015 г. в 1:38



j--

5,656

3

31

48

- 2 Я бы изменил последнюю `text` переменную `text="$color${@: 2}${code}0m"` таким образом, чтобы вся строка, за исключением параметра цвета, была окрашена. – [Шайрон Толедо](#) 1 окт. 2015 г. в 15:02
- 1 @tomazahlin просто добавьте `-e` к `echo`, как уже несколько раз упоминалось выше – [Artem Medvedev](#) 21 марта 2017 г. в 4:03
- 1 Как предложил Уилфред Хьюз, лучше использовать, `tput` так как он более переносим – работает в Bash и на macOS. Поэтому я на самом деле предлагаю использовать функцию Алирезы Мириана из этого ответа: [stack overflow .com/a/23006365/2693875](https://stackoverflow.com/a/23006365/2693875) – [Грег Дубицки](#) 7 июня 2020 г. в 18:17



Другие ответы уже дают отличные объяснения того, как это сделать. Чего мне все еще не хватало, так это хорошо организованного обзора цветовых кодов.

**38**

Статья в Википедии "[ANSI escape code](#)" очень полезна в этом. Однако, поскольку цвета часто можно настраивать и они выглядят по-разному в каждом терминале, я предпочитаю иметь функцию, которую можно вызвать в терминале. Для этой цели я создал следующие функции, чтобы показать таблицу цветов и напомнить мне, как их устанавливать (с расположением, вдохновленным статьей в Вики). Вы можете, например, загрузить их в свой `.bashrc/.zshrc` или поместить их где-нибудь как скрипт.



## 256 цветов

```
$ showcolors256 bg
Set foreground color: \033[38;5;NNNm
Set background color: \033[48;5;NNNm
Reset color & style: \033[0m
```

## 16 standard color codes:

000	001	002	003	004	005	006	007
008	009	010	011	012	013	014	015

## 6·6·6 RGB color codes:

016	017	018	019	020	021	022	023	024	025	026	027
052	053	054	055	056	057	058	059	060	061	062	063
088	089	090	091	092	093	094	095	096	097	098	099
124	125	126	127	128	129	130	131	132	133	134	135
160	161	162	163	164	165	166	167	168	169	170	171
196	197	198	199	200	201	202	203	204	205	206	207

028	029	030	031	032	033
064	065	066	067	068	069
100	101	102	103	104	105
136	137	138	139	140	141
172	173	174	175	176	177
208	209	210	211	212	213

034	035	036	037	038	039
070	071	072	073	074	075
106	107	108	109	110	111
142	143	144	145	146	147
178	179	180	181	182	183
214	215	216	217	218	219

040	041	042	043	044	045
076	077	078	079	080	081
112	113	114	115	116	117
148	149	150	151	152	153
184	185	186	187	188	189
220	221	222	223	224	225

046	047	048	049	050	051
082	083	084	085	086	087
118	119	120	121	122	123
154	155	156	157	158	159
190	191	192	193	194	195
226	227	228	229	230	231

## 24 grayscale color codes:

232	233	234	235	236	237	238	239	240	241	242	243
244	245	246	247	248	249	250	251	252	253	254	255

```
$ showcolors256 fg
Set foreground color: \033[38;5;NNNm
Set background color: \033[48;5;NNNm
Reset color & style: \033[0m

16 standard color codes:
 000  001  002  003  004  005  006  007
 008  009  010  011  012  013  014  015

6·6·6 RGB color codes:
 016  017  018  019  020  021    022  023  024  025  026  027
 052  053  054  055  056  057    058  059  060  061  062  063
 088  089  090  091  092  093    094  095  096  097  098  099
 124  125  126  127  128  129    130  131  132  133  134  135
 160  161  162  163  164  165    166  167  168  169  170  171
 196  197  198  199  200  201    202  203  204  205  206  207

 028  029  030  031  032  033    034  035  036  037  038  039
 064  065  066  067  068  069    070  071  072  073  074  075
 100  101  102  103  104  105    106  107  108  109  110  111
 136  137  138  139  140  141    142  143  144  145  146  147
 172  173  174  175  176  177    178  179  180  181  182  183
 208  209  210  211  212  213    214  215  216  217  218  219

 040  041  042  043  044  045    046  047  048  049  050  051
 076  077  078  079  080  081    082  083  084  085  086  087
 112  113  114  115  116  117    118  119  120  121  122  123
 148  149  150  151  152  153    154  155  156  157  158  159
 184  185  186  187  188  189    190  191  192  193  194  195
 220  221  222  223  224  225    226  227  228  229  230  231

24 grayscale color codes:
 232  233  234  235  236  237  238  239  240  241  242  243
 244  245  246  247  248  249  250  251  252  253  254  255
```

Сгенерировано этим скриптом bash/zsh:

```
function showcolors256() {
    local row col blockrow blockcol red green blue
    local showcolor=_showcolor256_${1:-bg}
    local white="\033[1;37m"
    local reset="\033[0m"

    echo -e "Set foreground color: \\\"\\033[38;5;${white}NNN${reset}\\\""
    echo -e "Set background color: \\\"\\033[48;5;${white}NNN${reset}\\\""
    echo -e "Reset color & style: \\\"\\033[0m\\\""
    echo

    echo 16 standard color codes:
```

```

for row in {0..1}; do
    for col in {0..7}; do
        $showcolor $(( row*8 + col )) $row
    done
    echo
done
echo

echo 6·6·6 RGB color codes:
for blockrow in {0..2}; do
    for red in {0..5}; do
        for blockcol in {0..1}; do
            green=$(( blockrow*2 + blockcol ))
            for blue in {0..5}; do
                $showcolor $(( red*36 + green*6 + blue + 16 )) $green
            done
            echo -n " "
        done
        echo
    done
    echo
done
echo
done

echo 24 grayscale color codes:
for row in {0..1}; do
    for col in {0..11}; do
        $showcolor $(( row*12 + col + 232 )) $row
    done
    echo
done
echo
}

function _showcolor256_fg() {
    local code=$( printf %03d $1 )
    echo -ne "\033[38;5;${code}m"
    echo -nE " $code "
    echo -ne "\033[0m"
}

function _showcolor256_bg() {
    if (( $2 % 2 == 0 )); then
        echo -ne "\033[1;37m"
    else
        echo -ne "\033[0;30m"
    fi
    local code=$( printf %03d $1 )
    echo -ne "\033[48;5;${code}m"
    echo -nE " $code "
    echo -ne "\033[0m"
}

```

## 16 цветов

```
$ showcolors16
\033[0;30m      \033[1;30m      \033[40m      \033[100m
\033[0;31m      \033[1;31m      \033[41m      \033[101m
\033[0;32m      \033[1;32m      \033[42m      \033[102m
\033[0;33m      \033[1;33m      \033[43m      \033[103m
\033[0;34m      \033[1;34m      \033[44m      \033[104m
\033[0;35m      \033[1;35m      \033[45m      \033[105m
\033[0;36m      \033[1;36m      \033[46m      \033[106m
\033[0;37m      \033[1;37m      \033[47m      \033[107m
```

Сгенерировано этим скриптом bash/zsh:

```
function showcolors16() {
    _showcolor "\033[0;30m" "\033[1;30m" "\033[40m" "\033[100m"
    _showcolor "\033[0;31m" "\033[1;31m" "\033[41m" "\033[101m"
    _showcolor "\033[0;32m" "\033[1;32m" "\033[42m" "\033[102m"
    _showcolor "\033[0;33m" "\033[1;33m" "\033[43m" "\033[103m"
    _showcolor "\033[0;34m" "\033[1;34m" "\033[44m" "\033[104m"
    _showcolor "\033[0;35m" "\033[1;35m" "\033[45m" "\033[105m"
    _showcolor "\033[0;36m" "\033[1;36m" "\033[46m" "\033[106m"
    _showcolor "\033[0;37m" "\033[1;37m" "\033[47m" "\033[107m"
}

function _showcolor() {
    for code in $@; do
        echo -ne "$code"
        echo -nE "    $code"
        echo -ne "\033[0m"
    done
    echo
}
```

Делиться Улучшить этот ответ

отредактировано 20 окт. 2021 г.

ответил 20 окт. 2021 г. в 15:23

Следовать

в 17:19

холодный ремонт

6,892 3 41 52

не возражаете, если я использую функции, которые вы разместили здесь, в плагине oh-my-zsh, предоставляющем функции colr и bgcolr? Это было бы, например echo -e "\$(colr 11 'yellow') normal \$(colr 124 'some red')". Смысл этого был бы в завершении zsh (скриншот). @coldfix ? - Касапер 5 окт. 2022 г. в 11:06

@Casaperg, без проблем, считайте это общественным достоянием. - [холодный ремонт](#) 5 окт. 2022 г. в 20:17

Используйте tput для вычисления цветовых кодов. Избегайте использования escape-кода ANSI (например, \E[31;1m для красного), так как он менее переносим. Например, Bash на OS X его не поддерживает.



```
BLACK=`tput setaf 0`
RED=`tput setaf 1`
GREEN=`tput setaf 2`
YELLOW=`tput setaf 3`
```



```

BLUE=`tput setaf 4`  

MAGENTA=`tput setaf 5`  

CYAN=`tput setaf 6`  

WHITE=`tput setaf 7`  
  

BOLD=`tput bold`  

RESET=`tput sgr0`  
  

echo -e "hello ${RED}some red text${RESET} world"

```

Делиться Улучшить этот ответ Следовать

ответил 4 апр. 2017 г. в 10:37

 Уилфред Хьюз  
30,7 тыс. 15 145  
199

Изящный способ изменить цвет только для одного объекта echo – определить такую функцию:

33

```

function coloredEcho(){  

    local exp=$1;  

    local color=$2;  

    if ! [[ $color =~ ^[0-9]$ ]] ; then  

        case $(echo $color | tr '[[:upper:]]' '[[:lower:]]) in  

            black) color=0 ;;  

            red) color=1 ;;  

            green) color=2 ;;  

            yellow) color=3 ;;  

            blue) color=4 ;;  

            magenta) color=5 ;;  

            cyan) color=6 ;;  

            white|*) color=7 ;; # white or invalid color  

        esac  

    fi  

    tput setaf $color;  

    echo $exp;  

    tput sgr0;  

}

```

Использование:

```
coloredEcho "This text is green" green
```

Или вы можете напрямую использовать цветовые коды, упомянутые в [ответе Дрю](#) :

```
coloredEcho "This text is green" 2
```

Делиться Улучшить этот ответ  
Следовать

отредактировано 23 мая 2017 г.  
в 12:26

ответил 11 апр. 2014 г. в 7:36

 Алиреза Мириан  
6,502 3 30 50

Сообщество Бот  
1 1

Если вы добавите -p в echo, то вы можете использовать его как встроенную раскраску echo "Red `coloredEcho "fox" red` jumps over the lazy dog" – [sobi3ch](#) 12

января 2017 г. в 17:23



28

Я нашел потрясающий ответ [Шакибы Мошири](#), когда искал информацию по этой теме... потом у меня возникла идея... и она превратилась в довольно приятную функцию, которую очень легко использовать 😊

Поэтому я должен ею поделиться 😊



<https://github.com/ppo/bash-colors>

**Использование:** \$(c <flags>) внутри echo -e или printf

Code	Style	Octal
-	Foreground	\033[3..
_	Background	\033[4..
k	Black	.....0m
r	Red	.....1m
g	Green	.....2m
y	Yellow	.....3m
b	Blue	.....4m
m	Magenta	.....5m
c	Cyan	.....6m
w	White	.....7m

Code	Style	Octal
B	Bold	\033[1m
U	Underline	\033[4m
F	Flash/blink	\033[5m
N	Negative	\033[7m
L	Normal (unbold)	\033[22m
0	Reset	\033[0m

## Примеры:

```
echo -e "$(c 0wB)Bold white$(c) and normal"
echo -e "Normal text... $(c r_yB)BOLD red text on yellow background... $(c _w)now on
white background... $(c 0U) reset and underline... $(c) and back to normal."
```

Делиться Улучшить этот ответ Следовать

ответил 4 апр. 2020 г. в 10:02



Паскаль Поллеунус

2,521 2 30 30



23

На этот вопрос уже отвечали много раз :-) но почему бы и нет.



Первое использование тут более переносимо в современных средах, чем ручное введение кодов ASCII через echo -E



Вот быстрая функция bash:

```
say() {
    echo "$@" | sed \
        -e "s/\\(\\(@\\
(red\\|green\\|yellow\\|blue\\|magenta\\|cyan\\|white\\|reset\\|b\\|u\\))\\)+\\)\\]\\{2\\}\\(.\\*
\\]\\]\\{2\\}\\}\\1\\4@reset/g" \
        -e "s/@red\\$(tput setaf 1)/g" \
        -e "s/@green\\$(tput setaf 2)/g" \
```

```

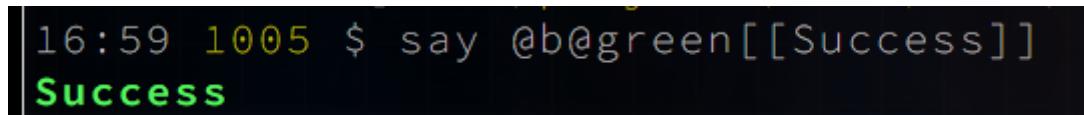
-e "s/@yellow/$(tput setaf 3)/g" \
-e "s/@blue/$(tput setaf 4)/g" \
-e "s/@magenta/$(tput setaf 5)/g" \
-e "s/@cyan/$(tput setaf 6)/g" \
-e "s/@white/$(tput setaf 7)/g" \
-e "s/@reset/$(tput sgr0)/g" \
-e "s/@b/$(tput bold)/g" \
-e "s/@u/$(tput sgr 0 1)/g"
}

```

Теперь вы можете использовать:

```
say @b@green[[Success]]
```

получить:



```
16:59 1005 $ say @b@green[[Success]]
Success
```

## Заметки о переносимости tput

Первый раз `tput(1)` исходный код был загружен в сентябре 1986 года.

`tput(1)` был доступен в семантике X/Open curses в 1990-х годах (стандарт 1997 года имеет семантику, указанную ниже).

Итак, это ( довольно ) повсеместно.

Делиться Улучшить этот ответ

отредактировано 13 окт. 2017 г. ответил 20 сен 2017 в 21:02

Следовать

в 23:39



Ахмед Масуд

22,1 тыс. 3 34 59

3 `tput` – это стандартный способ сделать это, где это совершенно не зависит от того, знаете ли вы возможности терминала. Если терминал не поддерживает заданную возможность, он изящно понизит уровень до того, что он может сделать, не выдавая сумасшедшие коды выхода. – [Ахмед Масуд](#) 5 окт. 2017 г. в 18:44

1 Я перестал использовать этот метод, так как он портит положение курсора в строках bash. Он будет случайным образом переноситься до конца строки и не вернется к началу строки при использовании `home` или клавиш со стрелками. Возвращение к неуклюжим ручным escape-кодам исправляет эту проблему. – [Редсандро](#) 22 окт. 2017 г. в 12:47

2 @Resandro – это потому, что вы используете его `$PS1` без `\[...\]` вокруг не пробельных частей? Продолжайте использовать маркеры Bash PS1 со строками `tput`. – [Тоби Спейт](#) 7 марта 2018 г. в 16:40

1 Мы можем установить позицию курсора, строку и столбец, используя похожую escape-последовательность ANSI. – [HBPM](#) 4 сен 2020 г. в 21:33

1 Эта функция порождает огромное количество подоболочек (по одной на каждый цвет) при каждом вызове. – [SergEd](#) 1 ноября 2023 г., 7:10



## Эмодзи

22

Единственное, что вы можете сделать, но не упомянуто в ответе, – это использовать эмодзи для раскрашивания выводимых данных!



```
echo 🟥: error message
echo 🟧: warning message
echo 🟩: ok status message
echo 🟪: action message
echo 📁: Or anything you like and want to recognize immediately by color
echo 💩: Or with a specific emoji
```



## Бонусная добавленная стоимость

Этот метод очень полезен, особенно когда ваш редактор исходного кода для скрипта поддерживает отображение Unicode. Тогда вы также можете увидеть красочный скрипт даже до его запуска и **непосредственно в исходном коде** ! :

56      echo 😢: All files are deleted!

Изображение файла скрипта внутри VSCode

**Примечание** : Возможно, вам придется передать Unicode эмодзи напрямую:

```
echo $'\U0001f972' // this emoji: 😢
```

Обратите внимание на **заглавные буквы U** для символов Unicode >= 10000.

**Также**, это бывает очень редко, но вам может потребоваться передать код следующим образом:

```
echo <0001f972>
```

Спасибо @joanis из комментариев за упоминание этого

Делиться Улучшить этот ответ

отредактировано 27 июня 2021 г.

в 20:25

Следовать

ответил 8 мая 2021 г. в 8:14  
 Mojtaba Hosseini  
 113 тыс. 41 306 302

3 Это интересная идея, но цвет эмодзи не отображается в моем терминале, они все преобразуются в текущий выводимый цвет. – [Джон](#) 18 июня 2021 г. в 16:24

Также, echo <0001f972> у меня не работает. В каком контексте этот синтаксис работает? Для символов Unicode <=ffff echo \$'\u1234' работает, но не для >=10000.

- [Джон](#) 18 июня 2021 г. в 16:26

Только что нашел ответ для `>=10000: echo $'\U0001f972'` с заглавной буквы `U`.  
 (вычислил это из [unix.stackexchange.com/a/280481/327696](https://unix.stackexchange.com/a/280481/327696), предположив, что bash и vim могут имитировать друг друга) - [Джон](#) 18 июня 2021 г. в 16:32 

 Если вы используете zsh или bash

 21

```
black() {
    echo -e "\e[30m${1}\e[0m"
}

red() {
    echo -e "\e[31m${1}\e[0m"
}

green() {
    echo -e "\e[32m${1}\e[0m"
}

yellow() {
    echo -e "\e[33m${1}\e[0m"
}

blue() {
    echo -e "\e[34m${1}\e[0m"
}

magenta() {
    echo -e "\e[35m${1}\e[0m"
}

cyan() {
    echo -e "\e[36m${1}\e[0m"
}

gray() {
    echo -e "\e[90m${1}\e[0m"
}

black 'BLACK'
red 'RED'
green 'GREEN'
yellow 'YELLOW'
blue 'BLUE'
magenta 'MAGENTA'
cyan 'CYAN'
gray 'GRAY'
```

[Попробуйте онлайн](#)

Делиться Улучшить этот ответ

Следовать

отредактировано 3 октября 2019

г. в 3:59

ответил 28 сен 2019 в 18:27



Viшал

20,5 тыс. 23 81 95



Мы можем использовать **24-битные RGB-настоящие цвета** как для текста, так и для фона!

21

```
ESC[38;2;<r>;<g>;<b>m /*Foreground color*/  
ESC[48;2;<r>;<g>;<b>m /*Background color*/
```



Пример красного текста и закрывающего тега:



```
echo -e "\e[38;2;255;0;0mHello world\e[0m"
```

Генератор:

[Показать фрагмент кода](#)

**24 бита:** поскольку графические карты «true color» с глубиной цвета от 16 до 24 бит стали обычным явлением, Xterm, Konsole от KDE, а также все терминалы на базе libvte (включая GNOME Terminal) поддерживают 24-битную настройку цвета переднего плана и фона

[https://en.wikipedia.org/wiki/ANSI\\_escape\\_code#24-bit](https://en.wikipedia.org/wiki/ANSI_escape_code#24-bit)

**Безопасно ли использовать его в моих скриптах?**

**Да!** 8- и 16-битные терминалы будут просто отображать как резервный цвет в диапазоне доступной палитры, сохраняя наилучшую контрастность, без разрывов!

Кроме того, никто не заметил полезности **реверсивного видео** в коде ANSI 7 .

Он остается читаемым на любых цветовых схемах терминала, на черном или белом фоне или в других необычных палитрах, путем замены цветов переднего плана и фона.

Пример красного фона, который подходит везде:

```
echo -e "\033[31;7mHello world\033[0m";
```

Вот как это выглядит при изменении встроенных схем терминала:

```
zen@book:~$ for i in {30..49};do echo -e "\033[$i;7mReversed color code $i\e[0m Hello world!";done
Hello world!
Reversed color code 31 Hello world!
Reversed color code 32 Hello world!
Reversed color code 33 Hello world!
Reversed color code 34 Hello world!
Reversed color code 35 Hello world!
Reversed color code 36 Hello world!
Reversed color code 37 Hello world!
Reversed color code 38 Hello world!
Reversed color code 39 Hello world!
Reversed color code 40 Hello world!
Reversed color code 41 Hello world!
Reversed color code 42 Hello world!
Reversed color code 43 Hello world!
Reversed color code 44 Hello world!
Reversed color code 45 Hello world!
Reversed color code 46 Hello world!
Reversed color code 47 Hello world!
Reversed color code 48 Hello world!
Reversed color code 49 Hello world!
zen@book:~$
```

Это циклический скрипт, используемый для gif-анимации.

```
for i in {30..49};do echo -e "\033[$i;7mReversed color code $i\e[0m Hello
world!";done
```

См.

[https://en.wikipedia.org/wiki/ANSI\\_escape\\_code#SGR\\_\(Select\\_Graphic\\_Rendition\)\\_parameters](https://en.wikipedia.org/wiki/ANSI_escape_code#SGR_(Select_Graphic_Rendition)_parameters)

Делиться Улучшить этот ответ

отредактировано 31 окт. 2019 г.

ответил 21 января 2018 г. в

в 4:42

10:51

Следовать



HBPM

12,6 тыс. 1 94 95

чтобы соответствовать zsh for i in {30..49};do echo -e "\033[\$i;7mReversed color code \$i\e[0m Hello world!";done – [админ](#) 11 сен 2021 г. в 3:32

Спасибо @k-five за этот ответ

20

```
declare -A colors
#curl www.bunlongheng.com/code/colors.png

# Reset
colors[Color_Off]='\033[0m'          # Text Reset

# Regular Colors
colors[Black]='\033[0;30m'            # Black
colors[Red]='\033[0;31m'              # Red
colors[Green]='\033[0;32m'             # Green
colors[Yellow]='\033[0;33m'            # Yellow
colors[Blue]='\033[0;34m'              # Blue
colors[Purple]='\033[0;35m'            # Purple
colors[Cyan]='\033[0;36m'              # Cyan
colors[White]='\033[0;37m'              # White

# Bold
colors[BBlack]='\033[1;30m'            # Black
colors[BRed]='\033[1;31m'              # Red
colors[BGreen]='\033[1;32m'             # Green
colors[BYellow]='\033[1;33m'            # Yellow
colors[BBBlue]='\033[1;34m'              # Blue
colors[BPurple]='\033[1;35m'             # Purple
colors[BCyan]='\033[1;36m'              # Cyan
```

```

colors[BWhite]='\033[1;37m'          # White

# Underline
colors[UBlack]='\033[4;30m'        # Black
colors[URed]='\033[4;31m'           # Red
colors[UGreen]='\033[4;32m'         # Green
colors[UYellow]='\033[4;33m'        # Yellow
colors[UBlue]='\033[4;34m'          # Blue
colors[UPurple]='\033[4;35m'        # Purple
colors[UCyan]='\033[4;36m'          # Cyan
colors[UWhite]='\033[4;37m'         # White

# Background
colors[On_Black]='\033[40m'         # Black
colors[On_Red]='\033[41m'            # Red
colors[On_Green]='\033[42m'          # Green
colors[On_Yellow]='\033[43m'         # Yellow
colors[On_Blue]='\033[44m'           # Blue
colors[On_Purple]='\033[45m'         # Purple
colors[On_Cyan]='\033[46m'           # Cyan
colors[On_White]='\033[47m'          # White

# High Intensity
colors[IBlack]='\033[0;90m'          # Black
colors[IRed]='\033[0;91m'             # Red
colors[IGreen]='\033[0;92m'           # Green
colors[IYellow]='\033[0;93m'          # Yellow
colors[IBlue]='\033[0;94m'            # Blue
colors[IPurple]='\033[0;95m'          # Purple
colors[ICyan]='\033[0;96m'            # Cyan
colors[IWhite]='\033[0;97m'           # White

# Bold High Intensity
colors[BIBlack]='\033[1;90m'          # Black
colors[BIRed]='\033[1;91m'             # Red
colors[BIGreen]='\033[1;92m'           # Green
colors[BIVYellow]='\033[1;93m'          # Yellow
colors[BIBlue]='\033[1;94m'            # Blue
colors[BIPurple]='\033[1;95m'          # Purple
colors[BICyan]='\033[1;96m'            # Cyan
colors[BIWhite]='\033[1;97m'           # White

# High Intensity backgrounds
colors[On_IBlack]='\033[0;100m'        # Black
colors[On_IRed]='\033[0;101m'           # Red
colors[On_IGreen]='\033[0;102m'          # Green
colors[On_IYellow]='\033[0;103m'         # Yellow
colors[On_IBlue]='\033[0;104m'           # Blue
colors[On_IPurple]='\033[0;105m'          # Purple
colors[On_ICyan]='\033[0;106m'           # Cyan
colors[On_IWhite]='\033[0;107m'          # White

color=${colors[$input_color]}
white=${colors[White]}
# echo $white

for i in "${!colors[@]}"
do
  echo -e "$i = ${colors[$i]}I love you$white"
done

```

## Результат

```
-bash ... ssh -i ~/keys/id_ssc-portal benu@172.19.242.31
Green = I love you
Black =
URed = I love you
BICyan = I love you
UPurple = I love you
BCyan = I love you
UYellow = I love you
BBrown = I love you
IRed = I love you
BGreen = I love you
Purple = I love you
On_IYellow = I love you
On_IGreen = I love you
On_IRed = I love you
On_IPurple = I love you
UBlack =
IBlue = I love you
IBlack = I love you
ICyan = I love you
OBrown = I love you
BYellow = I love you
On_IBlue = I love you
BIVYellow = I love you
BBBlack =
UCyan = I love you
BIPurple = I love you
Cyan = I love you
IRed = I love you
IBlack = I love you
IGreen = I love you
UGreen = I love you
IPurple = I love you
On_Cyan = I love you
BIBlack = I love you
IYellow = I love you
On_IWWhite = I love you
On_Blue = I love you
On_Yellow = I love you
On_Green = I love you
On_Purple = I love you
UWhite = I love you
Yellow = I love you
BIRed = I love you
IWWhite = I love you
Red = I love you
IGreen = I love you
Color_Off = I love you
On_IBlack = I love you
Slate = I love you
Blue = I love you
On_White = I love you
BIBlue = I love you
UBlue = I love you
BIBWhite = I love you
BIGreen = I love you
BPurple = I love you
White = I love you
[benu@benuportal]--[-]
```

Надеюсь, это [изображение](#) поможет вам выбрать цвет для вечеринки :D

Делиться Улучшить этот ответ

отредактировано 20 июня 2020 г.

ответил 28 апр. 2017 г. в 0:50

Следовать

в 9:12



Сообщество Бот

1 1



код-8

57,6 тыс. 116 380

651

Следует отметить, что для этого требуется bash v4. - [Синокс](#) 4 июня 2020 г. в 11:24

Вместо того, чтобы жестко кодировать escape-коды, специфичные для вашего текущего терминала, вам следует использовать tput .

18

Это мой любимый демо-скрипт:

```
#!/bin/bash

tput init

end=$(( $(tput colors)-1 ))
w=8
for c in $(seq 0 $end); do
    eval "$(\printf "tput setaf %3s  \"\$c\""; echo -n "\$_""
    [[ \$c -ge $(( w*2 )) ]] && offset=2 || offset=0
    [[ (((c+offset) % (w-offset))) -eq $(((w-offset)-1)) ]] && echo
done

tput init
```

```
tput setaf 8 tput setaf 9 tput setaf 10 tput setaf 11 tput setaf 12 tput setaf 13 tput setaf 14 tput setaf 15
tput setaf 22 tput setaf 23 tput setaf 24 tput setaf 25 tput setaf 26 tput setaf 27
tput setaf 28 tput setaf 29 tput setaf 30 tput setaf 31 tput setaf 32 tput setaf 33
tput setaf 34 tput setaf 35 tput setaf 36 tput setaf 37 tput setaf 38 tput setaf 39
tput setaf 40 tput setaf 41 tput setaf 42 tput setaf 43 tput setaf 44 tput setaf 45
tput setaf 46 tput setaf 47 tput setaf 48 tput setaf 49 tput setaf 50 tput setaf 51
tput setaf 52 tput setaf 53 tput setaf 54 tput setaf 55 tput setaf 56 tput setaf 57
tput setaf 58 tput setaf 59 tput setaf 60 tput setaf 61 tput setaf 62 tput setaf 63
tput setaf 64 tput setaf 65 tput setaf 66 tput setaf 67 tput setaf 68 tput setaf 69
tput setaf 70 tput setaf 71 tput setaf 72 tput setaf 73 tput setaf 74 tput setaf 75
tput setaf 76 tput setaf 77 tput setaf 78 tput setaf 79 tput setaf 80 tput setaf 81
tput setaf 82 tput setaf 83 tput setaf 84 tput setaf 85 tput setaf 86 tput setaf 87
tput setaf 88 tput setaf 89 tput setaf 90 tput setaf 91 tput setaf 92 tput setaf 93
tput setaf 94 tput setaf 95 tput setaf 96 tput setaf 97 tput setaf 98 tput setaf 99
tput setaf 100 tput setaf 101 tput setaf 102 tput setaf 103 tput setaf 104 tput setaf 105
tput setaf 106 tput setaf 107 tput setaf 108 tput setaf 109 tput setaf 110 tput setaf 111
tput setaf 112 tput setaf 113 tput setaf 114 tput setaf 115 tput setaf 116 tput setaf 117
tput setaf 118 tput setaf 119 tput setaf 120 tput setaf 121 tput setaf 122 tput setaf 123
tput setaf 124 tput setaf 125 tput setaf 126 tput setaf 127 tput setaf 128 tput setaf 129
tput setaf 130 tput setaf 131 tput setaf 132 tput setaf 133 tput setaf 134 tput setaf 135
tput setaf 136 tput setaf 137 tput setaf 138 tput setaf 139 tput setaf 140 tput setaf 141
tput setaf 142 tput setaf 143 tput setaf 144 tput setaf 145 tput setaf 146 tput setaf 147
tput setaf 148 tput setaf 149 tput setaf 150 tput setaf 151 tput setaf 152 tput setaf 153
tput setaf 154 tput setaf 155 tput setaf 156 tput setaf 157 tput setaf 158 tput setaf 159
tput setaf 160 tput setaf 161 tput setaf 162 tput setaf 163 tput setaf 164 tput setaf 165
tput setaf 166 tput setaf 167 tput setaf 168 tput setaf 169 tput setaf 170 tput setaf 171
tput setaf 172 tput setaf 173 tput setaf 174 tput setaf 175 tput setaf 176 tput setaf 177
tput setaf 178 tput setaf 179 tput setaf 180 tput setaf 181 tput setaf 182 tput setaf 183
tput setaf 184 tput setaf 185 tput setaf 186 tput setaf 187 tput setaf 188 tput setaf 189
tput setaf 190 tput setaf 191 tput setaf 192 tput setaf 193 tput setaf 194 tput setaf 195
tput setaf 196 tput setaf 197 tput setaf 198 tput setaf 199 tput setaf 200 tput setaf 201
tput setaf 202 tput setaf 203 tput setaf 204 tput setaf 205 tput setaf 206 tput setaf 207
tput setaf 208 tput setaf 209 tput setaf 210 tput setaf 211 tput setaf 212 tput setaf 213
tput setaf 214 tput setaf 215 tput setaf 216 tput setaf 217 tput setaf 218 tput setaf 219
tput setaf 220 tput setaf 221 tput setaf 222 tput setaf 223 tput setaf 224 tput setaf 225
tput setaf 226 tput setaf 227 tput setaf 228 tput setaf 229 tput setaf 230 tput setaf 231
tput setaf 232 tput setaf 233 tput setaf 234 tput setaf 235 tput setaf 236 tput setaf 237
tput setaf 238 tput setaf 239 tput setaf 240 tput setaf 241 tput setaf 242 tput setaf 243
tput setaf 244 tput setaf 245 tput setaf 246 tput setaf 247 tput setaf 248 tput setaf 249
tput setaf 250 tput setaf 251 tput setaf 252 tput setaf 253 tput setaf 254 tput setaf 255
```

Делиться Улучшить этот ответ

отредактировано 14 февр. 2020

г. в 6:25

Следовать

ответил 7 февр. 2020 г. в 6:27



Бруно Броноски

69,1 тыс. 14 168

150

Эти коды работают на моем компьютере с Ubuntu:

16

```
eric@dev:~$ echo -e "\x1B[31m foobar \x1B[0m"
foobar
eric@dev:~$ echo -e "\x1B[96m foobar \x1B[0m"
foobar
eric@dev:~$ echo -e "\x1B[01;96m foobar \x1B[0m"
foobar
eric@dev:~$ echo -e "\x1B[01;95m foobar \x1B[0m"
foobar
eric@dev:~$ echo -e "\x1B[01;94m foobar \x1B[0m"
foobar
eric@dev:~$ echo -e "\x1B[01;93m foobar \x1B[0m"
foobar
eric@dev:~$ echo -e "\x1B[01;92m foobar \x1B[0m"
foobar
eric@dev:~$ echo -e "\x1B[01;91m foobar \x1B[0m"
foobar
eric@dev:~$ echo -e "\x1B[01;90m foobar \x1B[0m"
foobar
eric@dev:~$ echo -e "\x1B[01;89m foobar \x1B[0m"
foobar
eric@dev:~$
```

```
echo -e "\x1B[31m foobar \x1B[0m"
echo -e "\x1B[32m foobar \x1B[0m"
echo -e "\x1B[96m foobar \x1B[0m"
echo -e "\x1B[01;96m foobar \x1B[0m"
```

```
echo -e "\x1B[01;95m foobar \x1B[0m"
echo -e "\x1B[01;94m foobar \x1B[0m"
echo -e "\x1B[01;93m foobar \x1B[0m"
echo -e "\x1B[01;91m foobar \x1B[0m"
echo -e "\x1B[01;90m foobar \x1B[0m"
echo -e "\x1B[01;89m foobar \x1B[0m"
echo -e "\x1B[01;36m foobar \x1B[0m"
```

Это напечатает буквы abcd разными цветами:

```
echo -e "\x1B[0;93m a \x1B[0m b \x1B[0;92m c \x1B[0;93m d \x1B[0;94m"
```

Для цикла:

```
for (( i = 0; i < 17; i++ ));  
do echo "$(tput setaf $i)This is ($i) $(tput sgr0)";  
done
```

```
eric@dev:~$ eric@dev /var/www/sandbox/eric $ for (( i = 0; i < 17; i++ )); do echo "$(tput setaf $i)This is ($i) $(tput sgr0)"; done

This is (1)
This is (2)
This is (3)
This is (4)
This is (5)
This is (6)
This is (7)
This is (8)
This is (9)
This is (10)
This is (11)
This is (12)
This is (13)
This is (14)
This is (15)

eric@dev /var/www/sandbox/eric $
```

Делиться Улучшить этот ответ

отредактировано 5 сентября 2014

ответил 5 сен 2014 в 18:40

Следовать

г. в 18:47



Эрик Лещински

152 тыс. 96 420 335

2 Кстати: это не сильно зависит от установленной конкретной версии Ubuntu, а от использования PuTTY! – [первобытные времена](#) 16 января 2015 г. в 14:50

## Для удобства чтения

12

Если вы хотите улучшить **читаемость** кода, вы можете echo сначала добавить строку, а затем цвет, используя sed :

`echo 'Hello World!' | sed $'s/World/\e[1m&\e[0m/'`

Делиться Улучшить этот ответ

отредактировано 20 июня 2020 г. в 9:12  
ответил 5 июля 2014 г. в 8:28

Следовать

Сообщество Бот  
1 1

Укер  
2,645 5 35 73

1 Мне очень нравится этот ответ! Можете ли вы объяснить, что такое \$ в команде sed?  
– Патрик 3 марта 2016 г. в 23:35

2 \$'<something>' для bash, а не sed. Он сообщает bash, что нужно обрабатывать \e как escape-последовательность и вставлять "escape"-символ. Обычно вы видите более простые формы, такие как \$'\t' или \$'\n', чтобы получить символ табуляции или новой строки, переданный команде. – дсз 10 окт. 2016 г. в 4:28

Вот что я в итоге использовал sed

12

```
echo "[timestamp] production.FATAL Some Message\n" \
"[timestamp] production.ERROR Some Message\n" \
"[timestamp] production.WARNING Some Message\n" \
"[timestamp] production.INFO Some Message\n" \
"[timestamp] production.DEBUG Some Message\n" | sed \
-e "s/FATAL/"$'\e[31m'"&"$'\e[m'"/" \
-e "s/ERROR/"$'\e[31m'"&"$'\e[m'"/" \
-e "s/WARNING/"$'\e[33m'"&"$'\e[m'"/" \
-e "s/INFO/"$'\e[32m'"&"$'\e[m'"/" \
-e "s/DEBUG/"$'\e[34m'"&"$'\e[m'"/"
```

Печатает так:

```
[timestamp] production.FATAL Some Message
[timestamp] production.ERROR Some Message
[timestamp] production.WARNING Some Message
[timestamp] production.INFO Some Message
[timestamp] production.DEBUG Some Message
```

Делиться Улучшить этот ответ Следовать

ответил 8 нояб. 2021 г. в 15:24

Оли Гирлинг  
723 9 17

На данный момент мой любимый ответ – colourEcho.

10

Чтобы разместить еще один вариант, вы можете проверить этот небольшой инструмент xcol

<https://ownyourbits.com/2017/01/23/colorize-your-stdout-with-xcol/>

вы используете его так же, как дгер, и он раскрасит свой stdin в разные цвета для каждого аргумента, например

```
sudo netstat -putan | xcol httpd sshd dnsmasq pulseaudio conky tor Telegram
firefox "[[:digit:]]+\.[[:digit:]]+\.[[:digit:]]+\.[[:digit:]]+" ":[[:digit:]]+"
"tcp." "udp." LISTEN ESTABLISHED TIME_WAIT
```

```
INSERT ~ sudo netstat -putan | xcol httpd sshd dnsmasq pulseaudio conky tor Telegram
firefox "[[:digit:]]+\.[[:digit:]]+\.[[:digit:]]+\.[[:digit:]]+" ":[[:digit:]]+"
"tcp." "udp." LISTEN ESTABLISHED TIME_WAIT
```

Active Internet connections (servers and established)					
Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	0.0.0.0:53	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:22	0.0.0.0:*	LISTEN
tcp	0	0	127.0.0.1:631	0.0.0.0:*	LISTEN
tcp	0	0	127.0.0.1:9050	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:4713	0.0.0.0:*	LISTEN
tcp	0	0	192.168.0.3:49060	8.39.54.57:443	ESTABLISHED
tcp	0	0	192.168.0.3:54304	149.154.167.91:80	ESTABLISHED
tcp	0	0	192.168.0.3:41534	176.34.244.212:80	ESTABLISHED
tcp	0	0	192.168.0.3:50904	149.154.167.91:443	ESTABLISHED
tcp	0	0	192.168.0.3:41996	54.149.244.33:443	ESTABLISHED
tcp	0	0	192.168.0.3:3853	192.168.0.5:2049	ESTABLISHED
tcp	0	0	192.168.0.3:41260	192.168.0.5:443	TIME_WAIT
tcp6	0	0	:::80	:::*	LISTEN
tcp6	0	0	:::53	:::*	LISTEN
tcp6	0	0	:::22	:::*	LISTEN
tcp6	0	0	:::631	:::*	LISTEN
tcp6	0	0	:::6600	:::*	LISTEN
tcp6	0	0	:::4713	:::*	LISTEN
udp	0	0	0.0.0.0:53	0.0.0.0:*	380/dnsmasq
udp6	0	0	:::53	:::*	380/dnsmasq

INSERT ~

Обратите внимание, что он принимает любое регулярное выражение, которое принимает sed.

Этот инструмент использует следующие определения

```
#normal=$(tput sgr0)
normal=$'\e[0m'
bold=$(tput bold)
red="$bold$(tput setaf 1)"
green=$(tput setaf 2)
fawn=$(tput setaf 3); beige="$fawn"
yellow="$bold$fawn"
darkblue=$(tput setaf 4)
blue="$bold$darkblue"
purple=$(tput setaf 5); magenta="$purple"
pink="$bold$purple"
darkcyan=$(tput setaf 6)
cyan="$bold$darkcyan"
gray=$(tput setaf 7)
darkgray="$bold$(tput setaf 0)
white="$bold$gray"

# normal text
# (works better sometimes)
# make colors bold/bright
# bright red text
# dim green text
# dark yellow text
# bright yellow text
# dim blue text
# bright blue text
# magenta text
# bright magenta text
# dim cyan text
# bright cyan text
# dim white text
# bold black = dark gray text
# bright white text
```

Я использую эти переменные в своих скриптах следующим образом:

```
echo "${red}hello ${yellow}this is ${green}coloured${normal}"
```

Делиться Улучшить этот ответ Следовать

ответил 24 января 2017 г. в  
8:23



нахопаркер  
1,706 19 14

Чтобы отобразить вывод сообщения другим цветом, можно сделать следующее:

10

```
echo -e "\033[31;1mYour Message\033[0m"
```

-Черный 0;30 Темно-серый 1;30

-Красный 0;31 Светло-красный 1;31

-Зеленый 0;32 Светло-зеленый 1;32

-Коричневый/Оранжевый 0;33 Желтый 1;33

-Синий 0;34 Светло-голубой 1;34

-Фиолетовый 0;35 Светло-фиолетовый 1;35

-Голубой 0;36 Светло-голубой 1;36

-Светло-серый 0;37 Белый 1;37

Делиться Улучшить этот ответ Следовать

ответил 19 авг. 2019 г. в 15:43



Амируш  
3,636 1 27 23

Я использую [это](#) для цветной печати.

10

```
#!/bin/bash
#-----
#Color picker, usage: printf $BLD$CUR$RED$BBLU'Hello World!'$DEF
#-----+-----+-----+
#      Text color      |      Background color      |      |
#-----+-----+-----+-----+
# Base color|Lighter shade| Base color      | Lighter shade      |
#-----+-----+-----+-----+
BLK='\e[30m'; blk='\e[90m'; BBLK='\e[40m'; bblk='\e[100m' #| Black
RED='\e[31m'; red='\e[91m'; BRED='\e[41m'; bred='\e[101m' #| Red
GRN='\e[32m'; grn='\e[92m'; BGRN='\e[42m'; bgrn='\e[102m' #| Green
YLW='\e[33m'; ylw='\e[93m'; BYLW='\e[43m'; bylw='\e[103m' #| Yellow
BLU='\e[34m'; blu='\e[94m'; BBLU='\e[44m'; bblu='\e[104m' #| Blue
MGN='\e[35m'; mgn='\e[95m'; BMGN='\e[45m'; bmgn='\e[105m' #| Magenta
CYN='\e[36m'; cyn='\e[96m'; BCYN='\e[46m'; bcyn='\e[106m' #| Cyan
WHT='\e[37m'; wht='\e[97m'; BWHT='\e[47m'; bwht='\e[107m' #| White
```

```

-----{ Effects }-----
DEF='\e[0m' #Default color and effects
BLD='\e[1m' #Bold\brighter
DIM='\e[2m' #Dim\darker
CUR='\e[3m' #Italic font
UND='\e[4m' #Underline
INV='\e[7m' #Inverted
COF='\e[?251' #Cursor Off
CON='\e[?25h' #Cursor On
-----{ Functions }-----
# Text positioning, usage: XY 10 10 'Hello World!'
XY () { printf "\e[$2;${1}H$3"; } #
# Print line, usage: line - 10 | line -= 20 | line 'Hello World!' 20 |
line () { printf -v _L %$2s; printf -- "${_L// /$1}"; } #
# Create sequence like {0..(X-1)}
que () { printf -v _N %$1s; _N=(${_N// / 1}); printf "${!_N[*]}"; } #
-----+

```

Все основные цвета установлены как переменные, а также есть несколько полезных функций: XY, line и que. Вставьте этот скрипт в один из своих и используйте все цветовые переменные и функции.

Делиться Улучшить этот ответ Следовать

ответил 14 февр. 2020 г. в 7:27

 Иван  
7,046 1 20 25

Вы можете «комбинировать» цвета и текстовый режим.

10

```

#!/bin/bash

echo red text / black background \Reverse\
echo "\033[31;7mHello world\e[0m";
echo -e "\033[31;7mHello world\e[0m";
echo

echo yellow text / red background
echo "\033[32;41mHello world\e[0m";
echo -e "\033[32;41mHello world\e[0m";
echo "\033[0;32;41mHello world\e[0m";
echo -e "\033[0;32;41mHello world\e[0m";
echo

echo yellow BOLD text / red background
echo "\033[1;32;41mHello world\e[0m";
echo -e "\033[1;32;41mHello world\e[0m";
echo

echo yellow BOLD text underline / red background
echo "\033[1;4;32;41mHello world\e[0m";
echo -e "\033[1;4;32;41mHello world\e[0m";
echo "\033[1;32;4;41mHello world\e[0m";
echo -e "\033[1;32;4;41mHello world\e[0m";
echo "\033[4;32;41;1mHello world\e[0m";
echo -e "\033[4;32;41;1mHello world\e[0m";
echo

```

```
red text / black background (Reverse)
\033[31;7mHello world\e[0m
Hello world

yellow text / red background
\033[32;41mHello world\e[0m
Hello world
\033[0;32;41mHello world\e[0m
Hello world

yellow BOLD text / red background
\033[1;32;41mHello world\e[0m
Hello world
\033[1;32;4;41mHello world\e[0m
Hello world
\033[4;32;41;1mHello world\e[0m
Hello world
```

Делиться Улучшить этот ответ Следовать

ответил 13 нояб. 2020 г. в  
16:12

F Вспышка  
192 2 6

Чтобы расширить [этот ответ](#), для ленивых из нас:

6

```
function echocolor() { # $1 = string
    COLOR='\033[1;33m'
    NC='\033[0m'
    printf "${COLOR}$1${NC}\n"
}

echo "This won't be colored"
echocolor "This will be colorful"
```

Делиться Улучшить этот ответ

отредактировано 23 мая 2017 г.

Следовать

в 12:02

ответил 23 сен 2015 в 16:21

 Песня  
16.5 тыс. 17 64 79

3 Не запрограммируйте хардкод на экранирование терминала. Используйте tput ; для этого он и нужен! – [Тоби Спейт](#) 23 сен 2015 в 16:25

@TobySpeight Хотя это может быть верно для поддержки нескольких платформ (теоретически), если автор считает, что это работает в его собственном мире, зачем не соглашаться и отговаривать других в похожем мире от использования этой техники? В данном случае я пробую подобное в Ubuntu 16.04 bash, и это работает. Как единственный пользователь на этой платформе, я нахожу этот ответ приемлемым. Я также буду использовать tput for sc и xc though (сохранить курсор, восстановить курсор). Хотя этот ответ называет меня «ленивым», его можно перефразировать как «практичный» или «прямо по делу». – [WinEunuuchs2Unix](#) 2 ноя 2019 в 21:18



Вам определено следует использовать `tput` вместо необработанных управляющих последовательностей ANSI.

6



Поскольку существует большое количество различных языков управления терминалом, обычно система имеет промежуточный уровень связи. Реальные коды ищутся в базе данных для текущего обнаруженного типа терминала, и выдает стандартизованные запросы к API или (из оболочки) к команде.



Одна из таких команд – `tput`. `tput` принимает набор аббревиатур, называемых именами возможностей, и любые параметры, если это необходимо, затем ищет правильные escape-последовательности для обнаруженного терминала в базе данных `terminfo` и выводит правильные коды (надеюсь, терминал их понимает).

из <http://wiki.bash-hackers.org/scripting/terminalcodes>

Тем не менее, я написал небольшую вспомогательную библиотеку под названием [bash-tint](#), которая добавляет еще один слой поверх `tput`, делая ее еще проще в использовании (*imho*):

Пример:

```
tint "white(Cyan(T)Magenta(I)Yellow(N)Black(T)) is bold(really) easy to use."
```

Даст следующий результат: **TINT is really easy to use.**

Делиться Улучшить этот ответ

отредактировано 24 апр. 2018 г.

ответил 23 апр. 2018 г. в 10:14

Следовать

в 15:41

ArtBIT



Тоби Спейт

29,6 тыс. 48 70 109



3,959 30 40



И вот что я использовал, чтобы просмотреть все комбинации и решить, какая из них звучит круто:

5



```
for (( i = 0; i < 8; i++ )); do
    for (( j = 0; j < 8; j++ )); do
        printf "$(tput setab $i)$(tput setaf $j)(b=$i, f=$j)$(tput sgr0)\n"
    done
done
```

Делиться Улучшить этот ответ

отредактировано 7 апр. 2017 г.

ответил 4 февр. 2017 г. в 14:03

Следовать

в 5:59

нет



триплии

186 тыс. 36 295 343



1,243 10 22



Я написал [swag](#) именно для этого.

2

Вы можете просто сделать



```
pip install swag
```



Теперь вы можете установить все команды escape в виде txt-файлов в указанное место назначения с помощью:

```
swag install -d <colorsdir>
```



Или еще проще через:

```
swag install
```

Который установит цвета в ~/.colors .

Либо вы используете их так:

```
echo $(cat ~/.colors/blue.txt) This will be blue
```

Или вот так, что я нахожу даже более интересным:

```
swag print -c red -t underline "I will turn red and be underlined"
```

Проверьте это на [ascinema](#) !

Делиться Улучшить этот ответ

отредактировано 26 авг. 2020 г.

ответил 18 февр. 2017 г. в

Следовать

в 21:30

21:30



[HoldOffHunger](#)

20,2 тыс. 11 114

142

бросает исключения в throw вас

1,706 14 17



Вдохновленный ответом @nachoparker, я добавил это в свой .bashrc :



```
#####
colours
source xcol.sh

### tput foreground
export tpfn=$'\e[0m' # normal
export tpfb=$(tput bold)

## normal colours
export tpf0=$(tput setaf 0) # black
export tpf1=$(tput setaf 1) # red
export tpf2=$(tput setaf 2) # green
export tpf3=$(tput setaf 3) # yellow
export tpf4=$(tput setaf 4) # blue
```



```
export tpf5=$(tput setaf 5) # magenta
export tpf6=$(tput setaf 6) # cyan
export tpf7=$(tput setaf 7) # white
# echo "${tpf0}black ${tpf1}red ${tpf2}green ${tpf3}yellow ${tpf4}blue
#${tpf5}magenta ${tpf6}cyan ${tpf7}white${tpfn}"

## bold colours
export tpf0b="$tpfb$tpf0" # bold black
export tpf1b="$tpfb$tpf1" # bold red
export tpf2b="$tpfb$tpf2" # bold green
export tpf3b="$tpfb$tpf3" # bold yellow
export tpf4b="$tpfb$tpf4" # bold blue
export tpf5b="$tpfb$tpf5" # bold magenta
export tpf6b="$tpfb$tpf6" # bold cyan
export tpf7b="$tpfb$tpf7" # bold white
# echo "${tpf0b}black ${tpf1b}red ${tpf2b}green ${tpf3b}yellow ${tpf4b}blue
#${tpf5b}magenta ${tpf6b}cyan ${tpf7b}white${tpfn}"
```

Это export позволяет мне использовать их tpf.. в скриптах Bash.

Делиться Улучшить этот ответ Следовать

ответил 29 сен 2020 в 16:44



djooharr

505 5 14

1 2 Следующий

 **Очень активный вопрос**. Заработайте 10 очков репутации (не считая бонуса ассоциации), чтобы ответить на этот вопрос. Требование репутации помогает защитить этот вопрос от спама и неответов.