

# Введение в коды терминалов (ANSI/VT100)

Коды терминала (управления) используются для выдачи определенных команд вашему терминалу. Это может быть связано с переключением цветов или позиционированием курсора, то есть со всем, что не может быть сделано самим приложением.

## Как это технически работает

Код управления терминалом - это специальная последовательность символов, которая печатается (как и любой другой текст). Если терминал понимает код, он не будет отображать последовательность символов, но выполнит некоторое действие. Вы можете распечатать коды с помощью простой `echo` команды.

**Примечание:** иногда я вижу коды, на которые ссылаются как "цвета Bash" (несколько "руководств по Bash" и т. Д.): Это совершенно неправильное определение.

## Команда `tput`


Поскольку существует большое количество различных языков управления терминалами, обычно система имеет промежуточный уровень связи. Реальные коды ищутся в базе данных **для текущего обнаруженного типа терминала**, и вы отправляете стандартизированные запросы в `API()` или (из командной оболочки) в команду.

Одна из этих команд `tput`. `Tput` принимает набор сокращений, называемых *именами возможностей*, и любые параметры, если это необходимо, затем ищет правильные escape-последовательности для обнаруженного терминала в `terminfo` базе данных и печатает правильные коды (надеюсь, терминал понимает).

## Коды

В этом списке я сосредоточусь на управляющих кодах ANSI / VT100 для наиболее распространенных действий - возьмите его в качестве краткого справочника. Документация вашего терминала или `terminfo` базы данных всегда является предпочтительным источником, когда что-то неясно! Кроме `tput` того, аббревиатуры обычно предназначены для выхода из ANSI!

Я перечислил только самые актуальные коды, конечно, любой терминал ANSI понимает гораздо больше! Но давайте сосредоточим обсуждение на общих сценариях оболочки 😊

Если я не смог найти соответствующий ANSI escape, вы увидите а  в качестве кода. Не стесняйтесь писать мне или исправлять это.

Коды ANSI всегда начинаются с символа ESC. (`ASCII()` 0x1B или восьмеричный 033) Это не входит в список, но **вам следует избегать прямого использования кодов ANSI - используйте `tput` команду!**

Все коды, которые можно использовать с `tput` , можно найти в `terminfo` (5). (по крайней мере, на OpenBSD) Смотрите `terminfo` OpenBSD (5) (<http://www.openbsd.org/cgi-bin/man.cgi?query=terminfo&apropos=0&sektion=5&manpath=OpenBSD+Current&arch=i386&format=html>) в разделе Возможностей. *Заглавное имя* - это код для использования с `tput`. Также предоставляется описание каждого кода.

## Общие полезные коды ASCII

Представление **клавиши Ctrl** просто связывает непечатаемые символы из `ASCII()`-кода 1 с печатными (буквенными) символами из `ASCII()`-кода 65 ("A"). Код `ASCII()` 1 будет `^A` (Ctrl-A), а код `ASCII()` 7 (BEL) будет `^G` (Ctrl-G). Это обычное представление (и метод ввода) и исторически происходит от одного из терминалов серии VT.

Имя	десятичные	восьмеричный	шестнадцатеричный	C-escape	Ctrl-клавиша	Описание
BEL	7	007	0x07	<code>\a</code>	<code>^G</code>	Звук терминала
BS	8	010	0x08	<code>\b</code>	<code>^H</code>	Вакансия
HT	9	011	0x09	<code>\t</code>	<code>^I</code>	Горизонтальная табуляция
LF	10	012	0x0A	<code>\n</code>	<code>^J</code>	Перевод строки (но не табуляция)
VT	11	013	0x0B	<code>\v</code>	<code>^K</code>	Вертикальная табуляция
FF	12	014	0x0C	<code>\f</code>	<code>^L</code>	Форматированная табуляция (табуляция строки)
CR	13	015	0x0D	<code>\r</code>	<code>^M</code>	Возврат каретки
ESC	27	033	0x1B	<code>&lt;none&gt;</code>	<code>^[</code>	Escape
DEL	127	177	0x7F	<code>&lt;none&gt;</code>	<code>&lt;none&gt;</code>	Удаление

## Управление курсором

ANSI	эквивалент terminfo	Описание
[ <X> ; <Y> H [ <X> ; <Y> f	cup <X> <Y>	X Y ⚠ Кажется, что ANSI использует 1-1 в качестве дома, а использует 0-0 tput
[ H	home	Переместите курсор в исходное положение (0-0)
7	sc	Сохранение текущего положения курсора
8	rc	Восстановить сохраненное положение курсора
⚠ скорее всего, обычный код, такой как \b	sub1	переместите влево на один пробел (backspace)
VT100 [ ? 25 l	civis	сделать курсор невидимым
VT100 [ ? 25 h	cvvis	сделать курсор видимым

## Удаление текста

ANSI	эквивалент terminfo	Описание
[ K [ 0 K	el	<b>Очистить строку</b> от текущего положения курсора <b>до конца</b> строки
[ 1 K	el1	<b>Очистить строку от начала</b> до текущего положения курсора
[ 2 K	el2 ⚠	<b>Очистить всю строку</b> (положение курсора не изменилось)

## Общие текстовые атрибуты

ANSI	эквивалент terminfo	Описание
[ 0 m	sgr0	Сбросить все атрибуты
[ 1 m	bold	Установите атрибут "яркий"

ANSI	эквивалент terminfo	Описание
[ 2 m	dim	Установите атрибут "dim"
[ 3 m	smso	Установите атрибут "выдающийся"
[ 4 m	установить, снять smul rmul ?	Установите атрибут "подчеркивание" (подчеркнутый текст)
[ 5 m	blink	Установите атрибут "мигать"
[ 7 m	rev	Установите атрибут "reverse"
[ 8 m	invis	Установите атрибут "скрытый"

## Раскраска переднего плана

ANSI	эквивалент terminfo	Описание
[ 3 0 m	setaf 0	Установите цвет <b>переднего</b> плана #0 - <b>черный</b>
[ 3 1 m	setaf 1	Установите цвет <b>переднего</b> плана # 1 - <b>красный</b>
[ 3 2 m	setaf 2	Установите цвет <b>переднего</b> плана # 2 - <b>зеленый</b>
[ 3 3 m	setaf 3	Установите цвет <b>переднего</b> плана # 3 - <b>желтый</b>
[ 3 4 m	setaf 4	Установите цвет <b>переднего</b> плана # 4 - <b>синий</b>
[ 3 5 m	setaf 5	Установите цвет <b>переднего</b> плана #5 - <b>пурпурный</b>
[ 3 6 m	setaf 6	Установите цвет <b>переднего</b> плана #6 - <b>голубой</b>
[ 3 7 m	setaf 7	Установите цвет <b>переднего</b> плана #7 - <b>белый</b>
[ 3 9 m	setaf 9	Установите цвет по <b>умолчанию</b> в качестве цвета переднего плана

# Раскраска фона

ANSI	эквивалент terminfo	Описание
[ 4 0 m	setab 0	Установить цвет <b>фона #0 - черный</b>
[ 4 1 m	setab 1	Установите цвет <b>фона # 1 - красный</b>
[ 4 2 m	setab 2	Установите цвет <b>фона # 2 - зеленый</b>
[ 4 3 m	setab 3	Установить цвет <b>фона # 3 - желтый</b>
[ 4 4 m	setab 4	Установить цвет <b>фона # 4 - синий</b>
[ 4 5 m	setab 5	Установить цвет <b>фона # 5 - пурпурный</b>
[ 4 6 m	setab 6	Установите цвет <b>фона # 6 - голубой</b>
[ 4 7 m	setab 7	Установить цвет <b>фона #7 - белый</b>
[ 4 9 m	setab 9	Установите цвет по <b>умолчанию</b> в качестве цвета фона

## Разные коды

### Экран сохранения / восстановления

Используемые возможности: `smcup` , `rmcup`

Вы, несомненно, уже сталкивались с программами, которые восстанавливают содержимое терминала после выполнения своей работы (например `vim` ). Это можно сделать с помощью следующих команд:

```
# сохранить, очистить экран
tput smcup
очистить

# ниже приведен пример "приложения"...
прочитайте -n1 -p "Нажмите любую клавишу, чтобы продолжить ..."
# пример "приложения" заканчивается здесь

# восстановить
tput rmcup
```

Эти функции требуют наличия определенных возможностей в вашем `termcap` / `terminfo`. Хотя `xterm` большинство его клонов (`rxvt` , `urxvt` , и т.д.) будут поддерживать инструкции, ваша операционная система может не включать ссылки на них в свой профиль `xterm` по умолчанию. (FreeBSD, в частности, попадает в эту категорию.) Если ``tput smcup``, похоже, ничего не делает для вас, и вы не хотите изменять свои системные данные `termcap` / `terminfo`, и вы ЗНАЕТЕ, что используете совместимое приложение `xterm`, вам может подойти следующее:

```
echo -e '\033[?47h' # сохранить экран
echo -e '\033[?47l' # восстановить экран
```

Некоторые программы также используют эти коды (через свои возможности `termcap`). Возможно, вы видели экран сохранения / восстановления в `less` , `vim` , `top` , `screen` и других. Некоторые из этих приложений могут также предоставлять параметры конфигурации для \* отключения \* такого поведения. Например, `less -X` для этого есть опция, которую также можно задать в переменной среды:

```
экспорт МЕНЬШЕ = X
меньше /путь/к /файлу
```

Аналогично, `vim` можно настроить, чтобы не "восстанавливать" экран, добавив следующее в свой `~/.vimrc` :

```
установите t_ti= t_te=
```

## Дополнительные цвета

Некоторые эмуляторы терминалов поддерживают дополнительные цвета. Наиболее распространенное расширение, используемое терминалами, совместимыми с `xterm`, поддерживает 256 цветов. Они могут быть сгенерированы с помощью `tput with setaf,b [0-255]` , когда `TERM` значение имеет `-256color` суффикс. Некоторые терминалы (<https://gist.github.com/XVilka/8346728#now-supporting-truecolour>) также поддерживают полные 24-битные цвета, и любой цветовой код X11 может быть записан непосредственно в специальную escape-последовательность. (Дополнительная информация (<https://gist.github.com/XVilka/8346728>)) Только несколько программ используют что-либо, кроме 256 цветов, и `tput` о них не знает. Цвета, превышающие 16, обычно применяются только к современным эмуляторам терминалов, работающим в графических средах.

Виртуальный терминал, реализованный в ядре Linux, поддерживает только 16 цветов, а обычная запись `terminfo` по умолчанию для `TERM=linux` определяет только 8. Иногда существует альтернативный `"linux-16color"`, на который вы можете переключиться, чтобы получить другие 8 цветов.

## Примеры Bash

### Жестко заданные цвета

```
printf '%b \n' 'Это \033[31mnot\033[39m разумно использовать \033[32m  
hardcoded ANSI\033[39m codes!'
```

## Цвета с использованием tput

---

### Непосредственно внутри echo:

```
echo "TPUT - это $ (tput setaf 2) приятная $ (tput setaf 9) и $ (tput  
setaf 5) удобная для пользователя $ (tput setaf 9) база данных возмож  
ностей терминала".
```

### С предустановленными переменными:

```
COL_NORM="$(tput setaf 9)"  
COL_RED="$(tput setaf 1)"  
COL_GREEN="$(tput setaf 2)"  
echo "Это $ {COL_RED} красный $ {COL_NORM} и $ {COL_GREEN} зеленый $  
{COL_NORM} - ты видел?"
```

## Разное

---

### ГЛАВНАЯ функция

```
home() {  
  # да, на самом деле не намного короче ;-)  
  tput home  
}
```

## Глупый, но приятный эффект

---





наполовину. doBash Функция использует целочисленную арифметику, но по-прежнему совместима с ksh93 (выполняется, например `bash ./mandelbrot`, для ее использования). Код с плавающей запятой только для ksh93 doksh почти в 10 раз быстрее doBash (таким образом, по умолчанию ksh shebang), но использует только те функции, которые не приводят к сбою синтаксического анализатора Bash.

```
#!/usr/bin/env ksh

# 16-цветной Мандельброт Чарльза Кука
# http://earth.gkhs.net/ccooke/shell.html
# Комбинированные вкусы Bash / ksh93 от Дэна Дугласа (ormaaaj)

функция doBash {
    набирает P Q X Y a b c i v x y
    для ((P=10**8, Q=P/100, X=320* Q/ cols, Y=210*Q/ lines, y=-105*Q, v=-
    220* Q, x=v; y
    <105*Q;x=v,y+=Y)); сделать
    для ((;x<4* P**2&&i++<99;a=((c=a)**2-b**2)/P + x,b=2*c*b/P +y)); дела
    ть :
        сделано
    colorBox $ ((i<99? i% 16:0))
    готово
, эхо
готово
}

функция doKsh {
    целое число i
    с плавающей точкой a b c x= 2.2 y =-1.05 X= 3.2/ cols Y= 2.1/строки
    , в то время
    как для ((a=b= i=0;(c = a)**2+b**2<=2&&i ++<99&&(a=a**2-b**2+x,b=2*c*
    b+y));); делать :
        Выполнено
        . colorBox $((i<99?i% 16:0))
        если ((x<1?!(x += X):(y +=Y, x=-2,2))); затем
    выведите
        ((y<1.05))
    я делаю
        :
    Выполнено
}

функция colorBox {
    (($ 1 == lastclr)) || printf %s "${colrs[lastclr=$1]}=$(tput setaf
    "$1")}"
    printf '\u2588'
}

unset -v lastclr
((cols=$(tput cols) -1, lines=$(tput lines)))
набор текста - ловушка colrs
'tput sgr0; echo' Выход
${KSH_VERSION+. doKsh} ${BASH_VERSION+doBash}
```

Гораздо более сложную версию от Roland Mainz можно найти здесь (<http://svn.nrubsig.org/svn/people/gisburn/scripts/mandelbrotset1.sh>)

## Обсуждение

Лукас Х., [2011/07/28 04:37 \(\)](#)

RE: ваше "Примечание: я не нашел кода для полного удаления текущей строки ("удалить строку" - это что-то другое!). Это может быть комбинация позиционирования курсора и стирания до конца строки ".

Попробуйте это: [ 2 K el2 Очистить всю строку

Джеймс, [2011/08/31 22:12 \(\)](#)

В таблице, показывающей

[ 3 9 m setaf 9 Установить цвет переднего плана по умолчанию

Описание "Установить цвет переднего плана по умолчанию" неоднозначно.

Эта фраза может означать либо то, что команды будут 1) сохранять значение указанного цвета в качестве значения цвета "по умолчанию", либо что 2) сохраненное значение цвета "по умолчанию" будет использоваться для переустановки текущего цвета переднего плана или фона на новое значение. Что это? В одном случае на экране может произойти видимое изменение. В другом случае на экране никогда не будет видимых изменений.

Как бы то ни было, некоторые люди создают файлы termcap, которые безвозмездно сбрасывают отображение на цвета "по умолчанию", что делает невозможным использование пользовательских цветов переднего плана и фона. Конечно, это просто подло и требует перезаписи файла termcap.

Кроме того, описания атрибутов "Dim", "Bright" и "Reverse" могут фактически указывать, что они должны делать. Например, что должно произойти при настройке как "Тусклый", так и "Яркий"? Или "Обратный" применяется как к цветам переднего плана, так и к цветам фона? Означает ли "Обратный" обмен цветами переднего плана и фона? Или установить какой-то "дополнительный" цвет для каждого из переднего плана и фона?

Эти "Описания", которые не описывают, бесполезны.

Джеймс

Константин, [2011/09/21 12:43 \(\)](#)

# распечатайте ярлыки для всех кодов ansi, ПРИМЕЧАНИЕ: пожалуйста, добавьте плюс плюс для утверждений!

```

ansi-тест ( )
{
для а в 0 1 4 5 7; выполнить
эхо "a= $a"
для (( f=0; f<=9; f ++ )) ; выполнить
для (( b=0; b<=9; b ++ )) ; выполнить
#echo -ne "f = $ f b = $ b"
echo -ne "\\ 033[$ {a};3 $ {f};4 $ {b} m"
echo -ne "\\033[033[${ a};3 $ {f};4$ {b} m"
echo -ne "\\033[0m "
сделано
эхо
сделано
эхо
сделано
эхо
}

```

Обри Бурк (<http://www.splashportal.net>), [2011/12/19 01:38 \(\)](#)

Привет,

Очень классный урок. Недавно я приобрел beagleboard XM, так что этот сайт - идеальное место для начала программирования последовательного порта.

И "глупые, но приятные эффекты" потрясающие. Мне это нравится!

Вот ссылка на "классную заставку для моего сайта". Это просто анимация Java... (открыть с помощью Java web start - jws)

файл:///home/aubrey/Downloads/circlesroller.jnlp  
(file:///home/aubrey/Downloads/circlesroller.jnlp)

С наилучшими пожеланиями.

Ян Шампера, [2011/12/21 11:35 \(\)](#), [2011/12/21 11:36 \(\)](#)

Привет,

Спасибо 😊

Я не думаю, что эта ссылка будет работать для кого-либо, кроме вас (файл :)



Билл Градвол, [2012/04/07 23:57 \(\)](#)

Это описывает вещи с точки зрения отображения. Что насчет клавиатуры? Как кто-то считывает коды с клавиатуры и выясняет, что пользователь нажал клавишу со стрелкой вверх, например?

Я заинтересован в этом для использования механизма чтения `bash -s -n 1` для ввода нажатий клавиш по 1 символу за раз, а затем попытаться выяснить, какую клавишу нажал пользователь. Например, стрелка вверх - это `\E[A`. Я хочу получить весь список возможных комбинаций символов, которые являются допустимыми для данной среды.

Утилита `infocmp` может сбрасывать `terminfo` для определенного объекта (`xterm`, `linux` и т. Д.), Но я не могу найти эквивалент для `keybaord`.

Ян Шампера, [2012/04/21 10:45 \(\)](#)

Очень хороший вопрос. Извините, я не могу ответить на это. Я думаю, что не существует таких вещей, как "стандартизированные" коды ключей.

Ruthard Baudach, [2013/04/03 19:35 \(\)](#), [2014/10/06 04:25 \(\)](#)

ну, просто используйте `read`!

`read` не только считывает ввод с клавиатуры, но и отображает его на терминале, в результате чего получаются коды клавиш, которые вы ищете.

Я использовал свои выводы для следующего скрипта на `python`: (извините, что не использовал `bash`)

```
#!/usr/bin/env python

импортируйте sys, termios, time

## Атрибуты шрифта ##
# выкл
выкл = '\x1b[0m' # выкл
по умолчанию = '\x1b[39m' # передний план по умолчанию
DEFAULT = '\x1b[49m' # фон по умолчанию
#
bd = '\ x1b [1m' # жирный
шрифт ft = '\ x1b [2m' # слабый
st = ' \ x1b [3m' # выделяющийся
ul = '\ x1b [4m' # подчеркнутый
bk = ' \ x1b [5m' # мигающий
rv = ' \ x1b [7m' #обратный
hd = '\ x1b [8m' # скрытый
nost = '\ x1b [23m' # нет
выделенного noul = ' \ x1b [24m' # нет подчеркнутого
nobk = ' \ x1b [25m' # нет мигания
norv = '\ x1b [27m' # нет обратного
# цвета
черный = '\x1b[30m'
ЧЕРНЫЙ = '\ x1b [40m'
красный = '\ x1b [31m'
КРАСНЫЙ = '\ x1b [41m'
зеленый = '\ x1b [32m'
ЗЕЛЕНый = '\ x1b [42m'
желтый = '\ x1b [33m'
ЖЕЛТЫЙ = '\ x1b [43m'
синий = '\ x1b [34m'
СИНИЙ = '\ x1b [44m'
пурпурный = '\ x1b [35m'
ПУРПУРНЫЙ = '\ x1b [45m'
голубой = '\ x1b [36m'
ГОЛУБОЙ = '\ x1b[46m'
белый = '\ x1b [37m'
БЕЛЫЙ = '\x1b[47m'
# светлые цвета
dgray = '\x1b[90m'
DGRAY = '\x1b[100m'
lred = '\x1b[91m'
LRED = '\x1b[101m'
lgreen = '\x1b[92m'
LGREEN = '\x1b[102 m'
lyellow = '\x1b[93m'
LYELLOW = '\x1b[103 м'
lblue = '\x1b[94m'
LBLUE = '\x1b[104 м'
lmagenta = '\x1b[95 м'
LMAGENTA = '\x1b[105 м'
lcyan = '\x1b[96m'
LCYAN = '\x1b[106m'
lgray = '\x1b[97m'
LGRAY = '\x1b[107m'
```

```

## 256 цветов ##
# \x1b[38;5;#m передний план, # = 0 - 255
# \x1b[48;5;#m фон, # = 0 - 255
## Истинный цвет ##
# \x1b[38;2; r; g;bm r = красный, g = зеленый, b = синий перед
ний план
# \x1b[48;2; r; g;bm r = красный, g = зеленый, b = синий фон

# -----
-----
# подготовьте настройки терминала
fd = sys.stdin.fileno()
старые настройки = termios.tcgetattr(fd)
новые настройки = termios.tcgetattr(fd)
новые настройки[3] &= ~termios.ICANON
new_settings[3] &= ~termios.EXO

# -----
-----
def очистить (что = 'экран'):
    '''
    функции стирания:
    что: экран => стереть экран и перейти к домашней
    строке => стереть строку и перейти к началу строки
    bos => стереть до начала экрана
    eos => стереть до конца экрана
    bol => стереть до начала строки
    eol => стереть до конца строки
    '''
    очистить = {
        'экран': '\x1b[2J \ x1b[H',
        'строка': '\x1b[2K \ x1b[G',
        'bos': '\x1b[1J',
        'eos': '\x1b[J',
        'bol': '\x1b[1K',
        'eol': '\x1b[K',
    }
    sys.стандартный вывод.написать (очистить [что])
    sys.стандартный вывод.промывка ()

# -----
-----
перемещение def (поз.):
    '''
    переместить курсор в pos
    pos = кортеж (x, y)
    '''
    x,y = pos
    sys.стандартный вывод.написать('\x1b[{};{}H'.формат (str(x),st
r(y)))
    sys.стандартный вывод.промывка ()

# -----
-----
def put(*аргументы):

```

```

'''
вывод текста на экран
кортеж в качестве первого аргумента указывает абсолютную позиц
ию для текста
, не изменяет положение курсора
args = список необязательных позиций, маркеров форматирования
и строк
'''

аргументы = список (аргументы)
если тип (аргументы [0]) == тип(()):
    x,y = args[0]
del args[0]
args.вставить(0, '\x1b[{};{}H'.format (str(x),str(y)))
аргументы.insert(0, '\x1b[s')
args.append('\x1b[u')
sys.стандартный вывод.написать (".join(аргументы))
sys.стандартный вывод.промывка ( )

# -----
-----
def write(*аргументы):
    '''
    выводит текст на экран
    кортеж в качестве первого аргумента задает относительную пози
цию к текущей позиции курсора
изменяет позицию курсора
args = список необязательных позиций, форматирующих маркеров и
строк
'''
    аргументы = список (аргументы)
    если тип (аргументы [0]) == тип(()):
        pos = []
        x,y = аргументы [0]
        если x > 0:
            pos.добавить('\x1b[{}A'.format (str(x)))
        elif x < 0:
            pos.добавить('\x1b[{}B'.format (abs(str(x))))
        если y > 0:
            pos.добавить('\x1b[{}C'.format (str(y)))
        если y < 0:
            pos.append('\x1b[{}D'.format(abs(str(y))))
        del args[0]
    args = pos + args
    sys.стандартный вывод.написать (".join(аргументы))
    sys.стандартный вывод.промывка ( )

# -----
-----
def getch():
    '''
    Получить символ.
    '''
    #
    попробуйте получить символ:
    termios.tcsetattr(fd, termios.TCSANOW, new_settings)

```

```
ch = sys.stdin.read(1)
наконец:
    termios.tcsetattr(fd, termios.TCSADRAIN, old_settings)
    # return
возврат ch
```

```
# -----
-----
```

```
если __name__ == '__main__':
```

```
очистить()
```

```
    esc_mode = False
```

```
    esc_string = "
```

```
    esc_codes = {
```

```
        '[A': 'Вверх',
```

```
        '[B': 'Вниз',
```

```
        '[C': 'Вправо',
```

```
        '[D': 'Влево',
```

```
        '[F': 'Конец',
```

```
        '[H': 'Pos1',
```

```
        '[2~': 'Ins',
```

```
        '[3~': 'Del',
```

```
        '[5~': 'PgUp',
```

```
        '[6~': 'PdDown',
```

```
        'OP': 'F1',
```

```
        'OQ': 'F2',
```

```
        'OR': 'F3',
```

```
        'OS': 'F4',
```

```
        '[15~': 'F5',
```

```
        '[17~': 'F6',
```

```
        '[18~': 'F7',
```

```
        '[19~': 'F8',
```

```
        '[20~': 'F9',
```

```
        '[21~': 'F10',
```

```
        '[23~': 'F11',
```

```
        '[24~': 'F12',
```

```
        '[29~': 'Приложения',
```

```
        '[34~': 'Победа',
```

```
        '[1;2A': 'S-Up',
```

```
        '[1;2B': 'S-Down',
```

```
        '[1;2C': 'S-Right',
```

```
        '[1;2D': 'S-Left',
```

```
        '[1;2F': 'S-End',
```

```
        '[1;2H': 'S-Pos1',
```

```
        '[2;2~': ' S-Ins',
```

```
        '[3;2~': ' S-Del',
```

```
        '[5;2~': 'S-PgUp',
```

```
        '[6;2~': ' S-PdDown',
```

```
        '[1;2P': 'S-F1',
```

```
        '[1;2Q': 'S-F2',
```

```
        '[1;2R': 'S-F3',
```

```
        '[1;2S': 'S-F4',
```

```
        '[15;2~': ' S-F5',
```

```
        '[17;2~': ' S-F6',
```

```
        '[18;2~': ' S-F7',
```



```
'[19;2~': 'S-F8',
'[20;2~': ' S-F9',
'[21;2~': ' S-F10',
'[23;2~': ' S-F11',
'[24;2~': 'S-F12',
'[29;2~': ' S-Приложения',
'[34;2~': ' S-Выигрыш',
'[1;3A': 'М-Вверх',
'[1;3B': 'М-Вниз',
'[1;3C': 'М-Right',
'[1;3D': 'М-Left',
'[1;3F': 'М-End',
'[1;3H': 'М-Pos1',
'[2;3~': 'М-Ins',
'[3;3~': ' М-Дел',
'[5;3~': ' М-PgUp',
'[6;3~': ' М-PdDown',
'[1;3P': 'М-F1',
'[1;3Q': 'М-F2',
'[1;3R': 'М-F3',
'[1;3S': 'М-F4',
'[15;3~': 'М-F5',
'[17;3~': ' М-F6',
'[18;3~': ' М-F7',
'[19;3~': ' М-F8',
'[20;3~': 'М-F9',
'[21;3~': ' М-F10',
'[23;3~': ' М-F11',
'[24;3~': ' М-F12',
'[29;3~': 'М-приложения',
'[34;3~': ' М-Выигрыш',
'[1;5A': 'С-Вверх',
'[1;5B': 'С-Вниз',
'[1;5C': 'С-Вправо',
'[1;5D': 'С-Left',
'[1;5F': 'С-End',
'[1;5H': 'С-Pos1',
'[2;5~': ' С-Ins',
'[3;5~': 'С-Del',
'[5;5~': ' С-PgUp',
'[6;5~': ' С-PdDown',
'[1;5P': 'С-F1',
'[1;5Q': 'С-F2',
'[1;5R': 'С-F3',
'[1;5S': 'С-F4',
'[15;5~': ' С-F5',
'[17;5~': ' С-F6',
'[18;5~': 'С-F7',
'[19;5~': ' С-F8',
'[20;5~': ' С-F9',
'[21;5~': ' С-F10',
'[23;5~': ' С-F11',
'[24;5~': ' С-F12',
'[29;5~': ' С-Приложения',
'[34;5~': ' С-Победа',
'[1;6A': 'S-C-Up',
```

```

'[1;6B': 'S-C-Down',
'[1;6C': 'S-C-Right',
'[1;6D': 'S-C-Left',
'[1;6F': 'S-C-End',
'[1;6H': 'S-C-Pos1',
'[2;6~': ' S-C-Ins',
'[3;6~': ' C-Си-Дел',
'[5;6~': ' S-C-PgUp',
'[6;6~': ' S-C-PdDown',
'[1;6P': 'S-C-F1',
'[1;6Q': 'S-C-F2',
'[1;6R': 'S-C-F3',
'[1;6S': 'S-C-F4',
'[15;6~': ' S-C-F5',
'[17;6~': ' C-C-F6',
'[18;6~': ' S-C-F7',
'[19;6~': ' S-C-F8',
'[20;6~': ' S-C-F9',
'[21;6~': 'S-C-F10',
'[23;6~': ' S-C-F11',
'[24;6~': ' S-C-F12',
'[29;6~': ' S-C-Приложения',
'[34;6~': 'S-C-Win',
'[1;7A': 'C-M-Up',
'[1;7B': 'C-M-Down',
'[1;7C': 'C-M-Right',
'[1;7D': 'C-M-Left',
'[1;7F': 'C-M-End',
'[1;7H': 'C-M-Pos1',
'[2;7~': ' C-M-Ins',
'[3;7~': 'C-M-Del',
'[5;7~': 'C-M-PgUp',
'[6;7~': ' C-M-PdDown',
'[1;7P': 'C-M-F1',
'[1;7Q': 'C-M-F2',
'[1;7R': 'C-M-F3',
'[1;7S': 'C-M-F4',
'[15;7~': ' C-M-F5',
'[17;7~': ' C-M-F6',
'[18;7~': ' C-M-F7',
'[19;7~': ' C-M-F8',
'[20;7~': ' C-M-F9',
'[21;7~': 'C-M-F10',
'[23;7~': ' C-M-F11',
'[24;7~': ' C-M-F12',
'[29;7~': ' C-M-Приложения',
'[34;7~': 'C-M-Win',
}
# 8 wäre S-C-M

ctrl_codes = {
  0: 'C-2',
  1: 'C-A',
  2: 'C-B',
  3: 'C-C',
  4: 'C-D',

```

```
5: 'C-E',
6: 'C-F',
7: 'C-G',
8: 'C-H',
9: 'C-I',
10: 'C-J',
11: 'C-K',
12: 'C-L',
13: 'C-M',
14: 'C-N',
15: 'C-O',
16: 'C-P',
17: 'C-Q',
18: 'C-R',
19: 'C-S',
20: 'C-T',
21: 'C-U',
22: 'C-V',
23: 'C-W',
24: 'C-X',
25: 'C-Y',
26: 'C-Z',
27: 'C-3',
29: 'C-5',
30: 'C-6',
31: 'C-7',
}
```

```
    при значении True:
переместить ((1,1))
очистить ("строка")
поставить ((1,1), зеленый, ': ', выкл.)
переместить ((1,3))
ch = getch()
если esc_mode:
    esc_string += ch
    # терминаторы строки esc
, если ch в ['A', 'B', 'C', 'D', 'F', 'H', 'P', 'Q',
'R', 'S', '~']:
    esc_mode = ложный
ход ((2,0))
очистить ("строка")
put((2,5), esc_codes[esc_string])
esc_string = "
elif ch == '\x1b':
    esc_mode = False
else:
    # режим esc
, если ch == '\x1b':
    esc_mode = True
    # ctrl
elif ord(ch) в ctrl_codes.keys():
переместить((2,0))
очистить ("строка")
put((2,5), ctrl_codes[ord(ch)])
```

```
перемещение ((2,0))  
put((2,3), str(ch))
```

Искрен Хаджинедев (<http://ikiji.com>), 2013/11/04 12:33 ()

Если вы используете X, вы можете получать коды клавиш с клавиатуры с помощью программы 'xev'; она открывает окно, которое печатает в терминале каждое событие (перемещение мыши, нажатие кнопки мыши, нажатие клавиши, освобождение клавиши и т. Д.). Я знаю, что опоздал более чем на год, но Google привел меня сюда, так что, надеюсь, кто-нибудь найдет это полезным. Приветствия.

Дж.К. Бенедикт, 2014/08/30 03:36 ()

Во-первых, спасибо за эту статью, поскольку я написал подпрограмму для различных \* nix и не-nix систем для анализа ANSI (как можно лучше). Суть в том, что я ОБОЖАЮ ЭТОТ ОБЗОР, особенно когда я сталкиваюсь с людьми, заинтересованными в том, чтобы максимально использовать bash и т. Д.

Второе - я могу внести свой вклад!

@Bill – Из bash используйте команду чтения. Я включил несколько ссылок для справки, но общая идея заключается в том, что его можно использовать для "Эй, введите что-то и нажмите enter", чтобы быть вложенным в условие цикла для "захвата" (это термин, на который вы захотите взглянуть) одиночных нажатий клавиш. Команда даже доходит до того, что дает "тайм-аут", если пользователь не нажимает ни одной клавиши!

[http://tldp.org/LDP/Bash-Beginners-Guide/html/sect\\_08\\_02.html](http://tldp.org/LDP/Bash-Beginners-Guide/html/sect_08_02.html)  
([http://tldp.org/LDP/Bash-Beginners-Guide/html/sect\\_08\\_02.html](http://tldp.org/LDP/Bash-Beginners-Guide/html/sect_08_02.html))

<http://www.unix.com/shell-programming-and-scripting/140231-bash-keypress-read-single-character.html> (<http://www.unix.com/shell-programming-and-scripting/140231-bash-keypress-read-single-character.html>)

и этот вид объединяет предыдущие ссылки в практический пример:

<http://top-scripts.blogspot.com/2011/01/blog-post.html> (<http://top-scripts.blogspot.com/2011/01/blog-post.html>)

НТН! -jkbs @xenfomation

Albert25, 2015/07/10 08:46 ()

Быстрый просмотр цветов переднего плана / фона:

```
для b в {0..7} 9; сделайте для f в {0..7} 9; сделайте для attr жирным шрифтом; сделайте echo -e " $ (tput setab $ b; tput setaf $ f; [ -n "$ attr" ] && tput $ attr)$ f НА $ b $ attr $ (tput sgr 0)"; сделано; сделано; сделано
```

Или то же самое в нескольких строках для удобства чтения:

```
для b в {0..7} 9; сделайте
для f в {0..7} 9; сделайте
для attr жирным шрифтом; сделайте
echo -e " $ (tput setab $ b; tput setaf $ f; [ -n "$ attr" ] && tput $ attr)$ f НА $ b $ attr $ (tput sgr0) "
сделано

, сделано, сделано
```