

## Bash (Русский)/Prompt customization (Русский)

[< Bash \(Русский\)](#)

**Состояние перевода:** На этой странице представлен

перевод статьи [Bash/Prompt customization](#). Дата

последней синхронизации: 10 июля 2021. Вы можете [помочь](#) синхронизировать перевод, если

в английской версии произошли [изменения](#).

### Ссылки по теме

- [Bash \(Русский\)](#)
- [Переменные окружения](#)
- [Git#Git prompt](#)

В Bash существует несколько приглашений командной строки, каждое из которых можно настроить на основе личных представлений об удобстве и эстетичности.

### Contents

- [1 Приглашения](#)
- [2 Техники](#)
  - [2.1 Escape-последовательности Bash](#)
  - [2.2 Escape-последовательности terminfo](#)
  - [2.3 Escape-последовательности ANSI](#)
  - [2.4 Встроенные команды](#)
  - [2.5 PROMPT\\_COMMAND](#)
  - [2.6 Escape-последовательности между вводом и выводом](#)

- [2.7 Настройка приглашения root](#)
- [3 Примеры](#)
- [3.1 Цвета](#)
- [3.2 Основные свойства](#)
- [3.3 Отображение кода выхода](#)
- [3.4 Позиционирование курсора](#)
- [3.4.1 Выравнивание по правому краю](#)
- [3.4.2 Произвольное позиционирование](#)
- [3.5 Настройка названия окна терминала](#)
- [4 Смотрите также](#)

## Приглашения

---

Bash имеет четыре *строки приглашения*, каждая из которых может быть настроена.

- **PS1** — основное приглашение, которое отображается перед каждой командой; по этой причине модифицируется чаще всего.
- **PS2** — второе приглашение, отображается, если команде требуются дополнительные данные для ввода (например, в случае многострочных команд).
- **PS3** — используется довольно редко. Отображается при работе встроенной команды Bash `select`, выводящей интерактивное меню. В отличие от остальных приглашений, не раскрывает **escape-последовательности**

**Bash.** Обычно все изменения применяются непосредственно в скрипте, содержащем `select`, а не в файле `.bashrc`.

- **PS4** — также используется редко. При отладке скриптов показывает уровни вложенности — первый символ приглашения повторяется столько раз, сколько на данный момент задействовано уровней.

Настройка конкретного приглашения подразумевает присваивание (обычно в файле `~/.bashrc`) необходимой строки в переменную, например:

```
PS2='> '
```

## Техники

---

Приглашение всегда можно задать строкой в явном виде, но существует ряд техник, позволяющих сделать его более динамичным и полезным.

### Escape-последовательности Bash

При выводе строки приглашения Bash ищет экранированные символом слэша символы (escape-последовательности) и конвертирует их в специальные строки. Например, `\u` превратится в имя пользователя, а `\A` — в текущее время. Таким образом, если переменной `PS1` присвоить `'\A \u $'`, то приглашение будет выглядеть как `17:35 пользователь $`.

Полный список escape-последовательностей можно найти в руководстве [bash\(1\) § PROMPTING](#) и в [справочнике Bash](#).

### Escape-последовательности terminfo

Помимо escape-последовательностей, которые понимает Bash, большинство терминалов также распознают специальные последовательности, которые влияют на терминал сам по себе, а не на печатаемые символы. Например, так можно изменить цвет строки символов, сдвинуть курсор в произвольную позицию или очистить экран. Эти последовательности могут быть довольно неудобными и варьируются от терминала к терминалу, поэтому они

задокументированы в базе данных `terminfo`. Чтобы увидеть, какие свойства поддерживает ваш терминал, выполните:

```
$ infocmp
```

Значение свойств можно найти в [terminfo\(5\)](#) по их названиям (часть перед `=`). Например, свойство `setaf` настраивает цвет шрифта для всего текста, который будет напечатан после него. Узнать escape-код свойства можно командой `tput`. Например,

```
$ tput setaf 2
```

выведет escape-последовательности для настройки зелёного цвета шрифта.

Примечание: Если команда `tput` не работает, убедитесь, что

значение `TERM` имеет верное значение для вашего терминала. Например,

если установлено значение `xterm` вместо `xterm-256color`, то `tput`

`setaf` будет работать только с номерами цветов 0-7.

На практике, чтобы использовать эти возможности в приглашении командной строки, можно использовать подстановку команд Bash и интерполяцию строк. Например:

```
GREEN="\[${tput setaf 2}\]"
```

```
RESET="\[${tput sgr0}\]"
```

```
PS1="${GREEN}my prompt${RESET}> "
```

```
my prompt>
```

Примечание: Руководство Bash рекомендует "обернуть" вывод `tput` в `\`

`[ \]`. Это поможет Bash правильно учитывать непечатаемые символы при

вычислении длины приглашения. При подстановке команд это не работает,

поэтому используйте **значения** `\1 \2`.

## Escape-последовательности ANSI

К сожалению, ANSI-последовательности могут отсутствовать в базе terminfo вашего терминала. Чаще всего это касается последовательностей для новейших возможностей вроде поддержки 256 цветов. В этом случае использовать `trut` не получится и придётся вводить escape-последовательности вручную.

Примеры escape-последовательностей можно найти в статье [Управляющие последовательности ANSI](#). Каждая последовательность начинается с литерала escape-последовательности, которую вы можете ввести с помощью escape-последовательности Bash `\e`. Например, `\e[48;5;209m` задаст персиковый цвет фона (если есть поддержка 256 цветов), а `\e[2;2H` сдвинет курсор в левый верхний угол экрана.

В случаях, когда escape-последовательности Bash не поддерживаются (как в приглашении PS3), их можно добавить командой `printf`:

```
ESC=$(printf "\e")
PEACH="$ESC[48;5;209m"
```

## Встроенные команды

Если вы хотите добавить вывод какой-нибудь команды в приглашение, то используйте подстановку команд (command substitution). Например, чтобы добавить величину свободной памяти к приглашению попробуйте что-то вроде:

```
PS1="$ (awk '/MemFree/{print $2}' /proc/meminfo) prompt > "
53718 prompt >
53718 prompt >
53718 prompt >
```

Как видно, это работает не совсем корректно — значение памяти всегда одно и то же! Причина — команда выполняется только один раз при первой настройке PS1. Необходимо предотвратить подстановку либо экранированием символа `$`, либо определением строки в одиночных кавычках — в обоих случаях подстановка будет производиться каждый раз при настоящем отображении приглашения:

```
PS1="\$(awk '/MemFree/{print \$2}' /proc/meminfo) prompt > "  
# или  
PS1='$(awk "/MemFree/{print \$2}" /proc/meminfo) prompt > '
```

Если команды сделали приглашение слишком длинным, для лучшей читабельности можно вынести их в функцию:

```
free_mem()  
{  
    awk '/MemFree/{print $2}' /proc/meminfo  
}  
  
PS1='$(free_mem) prompt > '
```

Примечание: В подстановочных функциях можно использовать escape-последовательности terminfo/ANSI, но **не** последовательности Bash. В частности, `\[ \]` не будет работать при обрамлении ими строки с непечатаемыми символами. Вместо этого используйте восьмеричные экранированные последовательности `\001` и `\002` (например, в командах `printf` или `echo -e`).

## PROMPT\_COMMAND

Переменной `PROMPT_COMMAND` можно присвоить произвольную команду, которая будет выполняться непосредственно перед выводом PS1. Это позволяет создавать довольно мощные эффекты. Например, можно

переназначить PS1 на основе некоторых условий, или выполнить какие-то действия с историей Bash при выполнении любой команды.

Важно: `PROMPT_COMMAND` не должна использоваться для вывода символов

непосредственно в приглашение. Символы, напечатанные вне PS1, не

учитываются Bash, что может привести к неправильному

позиционированию курсора и обычных символов. Либо

используйте `PROMPT_COMMAND` для задания PS1, либо изучите

рекомендации в разделе [#Встроенные команды](#).

Совет: Если `PROMPT_COMMAND` стала слишком сложной, **bash-**

**preexec** (реализация хук-функций `preexec` и `precmd` **Zsh** для Bash)

может упростить работу с ней.

## Escape-последовательности между вводом и выводом

Свойства вводимого текста можно изменить, "забыв" отключить свойства в конце PS1. Например, если вставить `tput blink` в конец PS1, то вводимые команды будут мерцать. Тем не менее, этот эффект также перейдёт и на вывод команды, поскольку свойства не отключаются при нажатии `Enter`.

Чтобы вставить escape-последовательность после ввода, но перед началом вывода, можно перехватить (trap) Bash-сигнал `DEBUG`, который посылается перед выполнением каждой команды:

```
$ trap 'tput sgr0' DEBUG
```

## Настройка приглашения root

Для удобства можно сделать приглашение командной строки root-пользователя визуально отличным от обычного (возможно, мерцающий

красный цвет?). Настройка приглашения производится как обычно, но в домашнем каталоге суперпользователя, `/root`. Начните с копирования шаблонов `/etc/skel/.bash_profile` и `/etc/skel/.bashrc` в каталог `/root`, после чего внесите в файл `/root/.bashrc` необходимые изменения.

## Примеры

---

### Цвета

Совет: Вывод `infocmp` содержит доступное для `tput` количество цветов, например — `colors#8`.

Увидеть все цвета вашего терминала можно с помощью простого цикла (замените `setab` на `setaf`, если нужен цвет текста, а не фона):

```
for C in {0..255}; do
    tput setab $C
    echo -n "$C "
done
tput sgr0
echo
```

Если это не работает (причём установлено **правильное значение TERM**), протестируйте вручную разные последовательности:

```
# стандартные цвета
for C in {40..47}; do
    echo -en "\e[${C}m$C "
done
# цвета высокой интенсивности
for C in {100..107}; do
    echo -en "\e[${C}m$C "
done
# 256 цветов
for C in {16..255}; do
    echo -en "\e[48;5;${C}m$C "
done
echo -e "\e(B\e[m"
```



Аналогичные значения для текста (не фона): стандартные — 30..37, высокая интенсивность — 90..97, а для 256 цветов замените 48 на 38.

## Основные свойства

Следующие **свойства terminfo** будут полезны при настройке приглашения и поддерживаются во многих терминалах. **#1** и **#2** необходимо заменить на числовые аргументы.

Свойство	Escape-последовательность	Описание
Свойства текста		
blink	\E[5m	мерцающий текст вкл
bold	\E[1m	полужирный текст вкл
dim	\E[2m	тусклый текст вкл
rev	\E[7m	обратное отображение вкл (текст/фон меняются цветами)
sitm	\E[3m	курсив вкл
ritm	\E[23m	курсив выкл
smso	\E[7m	выделение текста вкл
rmso	\E[27m	выделение текста выкл
smul	\E[4m	подчёркивание вкл
rmul	\E[24m	подчёркивание выкл
setab #1	\E[4#1m	задать цвет фона #1 (0-7)
setaf #1	\E[3#1m	задать цвет текста #1 (0-7)
sgr0	\E(B\E[m	отключить все атрибуты текста
Перемещение курсора		
sc	\E7	сохранить позицию курсора
rc	\E8	вернуть курсор в сохранённую позицию
clear	\E[H\E[2J	очистить экран и переместить курсор в левый верхний угол
cuu #1	\E[#1A	переместить курсор вверх на #1 строк
cud #1	\E[#1B	переместить курсор вниз #1 строк
cuf #1	\E[#1C	переместить курсор вправо #1 столбцов
cub #1	\E[#1D	переместить курсор влево #1 столбцов
home	\E[H	переместить курсор в левый верхний угол окна
hpa #1	\E[#1G	переместить курсор в столбец #1
vpa #1	\E[#1d	переместить курсор в строку #1, первый столбец
cup #1 #2	\E[#1;#2H	переместить курсор в строку #1, столбец #2
Удаление символов		
dch #1	\E#1P	удалить #1 символов (аналогично нажатию клавиши

		backspace)
dl #1	\E#1M	удалить #1 строк
ech #1	\E#1X	стереть #1 символов (без перемещения курсора)
ed	\E[J	очистить до нижнего края экрана
el	\E[K	очистить до конца строки
el1	\E[1K	очистить до начала строки

## Отображение кода выхода

Тем же приёмом, как в случае **встроенных команд**, можно отложить интерполяцию специальной переменной Bash вроде `$?`. Следующие приглашения будут содержать код выхода предыдущей команды:

```
PS1="\$? > "
# или
PS1=' $? > '
```

```
0 > true
```

```
0 > false
```

```
1 >
```

Это можно сделать с помощью условных выражений и функций:

```
exitstatus()
{
    if [[ $? == 0 ]]; then
        echo ':)'
    else
        echo 'D:'
    fi
}
PS1='${exitstatus} > '
```

```
:) > true
```

```
:) > false
```

```
D: >
```

## Позиционирование курсора

Курсор можно перемещать по экрану во время нахождения "внутри" приглашения PS1, чтобы разные части приглашения появлялись в разных местах. Важный момент — после всех перемещений и вывода символов в любых местах экрана курсор необходимо вернуть в исходную позицию. Это можно сделать с помощью свойств `sc` и `rc`, которые сохраняют и восстанавливают позицию курсора соответственно. Общая схема приглашения, содержащего перемещения курсора:

```
PS1="\[$(tput sc; перемещение курсора) работа с курсором $(tput rc)\] работа с курсором после возврата"
```

Весь блок с перемещениями курсора обернут в `\[ \]`, чтобы Bash не учитывал непечатаемые символы как часть приглашения.

### Выравнивание по правому краю

Простейший способ напечатать текст у правого края экрана — использовать `printf`:

```
rightprompt()  
{  
    printf "%*s" $COLUMNS "right prompt"  
}  
  
PS1='\[$(tput sc; rightprompt; tput rc)\]left prompt > '
```

```
left prompt > right prompt
```

Здесь задано поле `%*s` переменной длины с выравниванием по правому краю. Размер поля равен текущему количеству столбцов в терминале (`$COLUMNS`).

### Произвольное позиционирование

Свойство `cup` перемещает курсор в конкретную позицию экрана, например, `tput cup 20 5` переместит курсор на строку 20, столбец 5

(координаты 0 0 обозначают верхний левый угол). `cuu`, `cud`, `cuf` и `cub` (вверх, вниз, вперёд, назад) перемещают курсор относительно текущей позиции. Например, `tput cuf 10` переместит курсор на 10 символов вправо. В аргументах можно использовать переменные `LINES` и `COLUMNS`, если требуется переместить курсор относительно нижнего и правого краёв окна. Например, перемещение на 10 строк и 5 столбцов от правого нижнего угла:

```
$ tput cup $((LINES - 11)) $((COLUMNS - 6))
```

## Настройка названия окна терминала

Название окна терминала можно настроить так же, как и приглашение: выводом escape-последовательностей в оболочке. Часто пользователи встраивают настройки названия окна в своё приглашение. Технически это возможно у `xterm`, но и другие современные терминалы её поддерживают. В этом случае используют последовательности `ESC]2;новое названиеBEL`, где `ESC` и `BEL` — символы escape (выход) и bell (сигнал).

С **последовательностями Bash** приглашение с встроенным названием окна будет иметь вид:

```
PS1='\[\e]2;новое название\a\]prompt > '
```

Само собой, строка названия окна может включать вывод **встроенных команд** или переменные вроде `$PWD`, так что она может перенастраиваться после каждой команды.

## Смотрите также

---

- Примеры и скриншоты на теме форума: [What's your PS1?](#) (доступно только после входа)
- [Файл /etc/bash/bashrc для Gentoo](#); см. также `gentoo-bashrc`<sup>AUR</sup>
- `tput(1)`
- [Справка по tput на сайте bash-hackers.org](#)

- [Цвета и перемещение курсора с tput](#)
- [Приглашение Bash HOWTO](#)
- [Коллекция примеров приглашений от Giles Orr](#)
- [Советы Bash: цвета и форматирование](#)
- [Liquid Prompt — полезное адаптивное приглашение для Bash & zsh](#)
- [Bash POWER PROMPT](#)
- [Wikipedia:ru:Управляющие последовательности ANSI](#)
- [Руководство GNU Bash: управление приглашением](#)

Categories:

- [Eye candy \(Русский\)](#)
- [Command-line shells \(Русский\)](#)