

**Олег Самойлов** @splarv

Пользователь

11,0

карма

24,1

рейтинг



Профиль

1

Публикации

10

Комментарии

0

Избранное

1

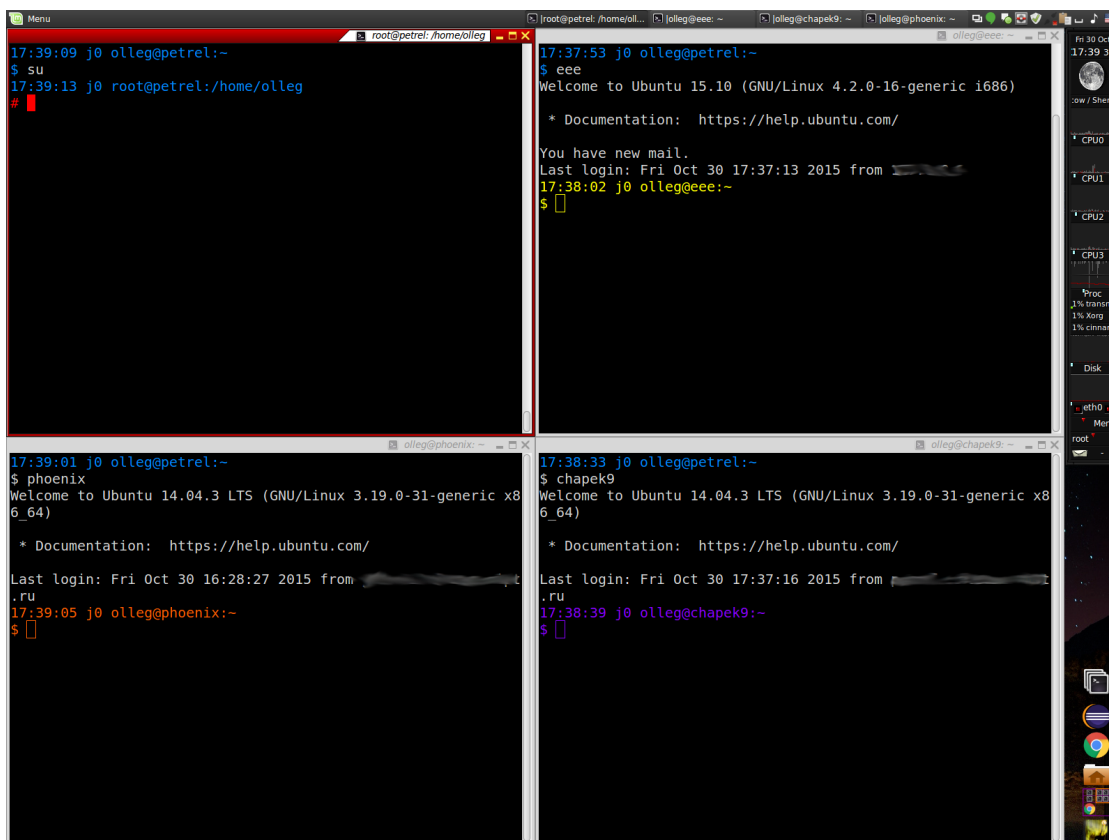
Подписчики

вчера в 12:49

Разноцветные терминалы

из песочницы

Системное администрирование*, Оболочки*, Настройка Linux*, *nix*



В этой публикации я расскажу о некоторых трюках, которые украсят будни любого системного администратора Linux (и не только). Все они связаны с переменной PS1 оболочки bash. Переменная PS1 определяет, как будет выглядеть приглашение для ввода новых команд. И каждый пользователь может переопределять её как пожелает, например, в файле ~/.bashrc (который выполняется при запуске bash и используется для в том числе для конфигурации).

Для начала рассмотрим простой вариант, мой любимый формат командной строки.

```
PS1='\t j\j \u@h:\w\n\S '
```

Результат будет вот такой:

```
17:42:46 j0 olleg@petrel:~  
$
```

Это обычное использование переменной PS1, но если бы я не начал с этого — рассказ был бы неполным.

Обычно в переменной PS1 с помощью специальных последовательностей символов определяют формат приглашения для ввода команд. Подробный список этих последовательностей можно почитать в документации к bash, в данном примере:

- `\t` — вывод «текущего времени», на самом деле это получается время завершения выполнения предыдущей команды, удобно когда перед глазами.
- `j\j` — выводит символ j и после него количество запущенных job, т.е. процессов в фоне. Это тоже удобно иметь перед глазами чтобы случайно про них не забыть когда соберешься закрыть терминал.
- `\u@\h` — имя пользователя и название сервера. Если работаете с несколькими серверами через удаленные терминалы — чтобы не путаться.
- `\w` — после двоеточия — рабочая директория.
- `\n` — поскольку строка получилась хоть и информативной (что-то вроде статус бара), но длинной, то приглашаем вводить команды с новой строки, а эта верхняя строка будет наглядно отделять от результата работы предыдущей команды.
- `\$` — на новой строке будет выводиться символ либо \$ для обычного пользователя либо # для root'a и отделив его пробелом можно приглашать вводить новую команду.

Казалось бы, чего еще желать... Но дальше будет интереснее. Дело в том, что с помощью специальных управляющих символов можно задавать цвет выводимого текста, цвет курсора и даже переопределять title bar у таких графических терминалов, как Gnome2. И, на мой взгляд, довольно удобно когда цветом отделяются терминалы запущенные на различных серверах. Для меня каждый сервер ассоциируется с каким-то цветом и в этот цвет мы будем красить командную строку и курсор на каждом сервере.

У меня .bashrc разделен на два файла, в самом .bashrc содержится общий код для всех серверов, а в .bash_local — уникальные для этого сервера настройки командной строки. .bash_local я буду вставлять в .bashrc специальной директивой. Начнем с .bash_local. В контексте данной статьи там у меня будут две строчки, которые определяют цвет этого сервера:

```
# .bash_local
# change cursor and prompt color
cursor_color='#0087FF'
prompt_color='33'
```

Просто заносу коды цвета в переменные. Но, как вы заметили, что способ задания цвета для курсора и для текста командной строки — разный. Почему-то так исторический получилось. Чтобы понять, какой цвет каким кодом кодируется, есть подходящая картинка.

	000000	005f00	008700	00af00	00d700	00ff00	5fff00	5fd700	5faf00	5f8700	5f5f00	5f0000
16	22	28	34	40	46	82	76	70	64	58	52	
	00005f	005f5f	00875f	00af5f	00d75f	00ff5f	5fff5f	5fd75f	5faf5f	5f875f	5f5f5f	5f005f
17	23	29	35	41	47	83	77	71	65	59	53	
	000087	005f87	008787	00af87	00d787	00ff87	5fff87	5fd787	5faf87	5f8787	5f5f87	5f0087
18	24	30	36	42	48	84	78	72	66	60	54	
	0000af	005faf	0087af	00afaf	00d7af	00ffaf	5fffaf	5fd7af	5fafaf	5f87af	5f5faf	5f00af
19	25	31	37	43	49	85	79	73	67	61	55	
	0000d7	005fd7	0087d7	00afd7	00d7d7	00ffd7	5ffd7	5fd7d7	5fafd7	5f87d7	5f5fd7	5f00d7
20	26	32	38	44	50	86	80	74	68	62	56	
	0000ff	005fff	0087ff	00afff	00d7ff	00ffff	5fffff	5fd7ff	5fafff	5f87ff	5f5fff	5f00ff
21	27	33	39	45	51	87	81	75	69	63	57	
	8700ff	875fff	8787ff	87afff	87d7ff	87ffff	afffff	afd7ff	afafff	af87ff	af5fff	af00ff
93	99	105	111	117	123	159	153	147	141	135	129	
	8700d7	875fd7	8787d7	87afd7	87d7d7	87ffd7	afffd7	afd7d7	afafd7	af87d7	af5fd7	af00d7
92	98	104	110	116	122	158	152	146	140	134	128	
	8700af	875faf	8787af	87afaf	87d7af	87ffaf	afffaf	afd7af	afafaf	af87af	af5faf	af00af
91	97	103	109	115	121	157	151	145	139	133	127	
	870087	875f87	878787	87af87	87d787	87ff87	afff87	afd787	afaf87	af8787	af5f87	af0087
90	96	102	108	114	120	156	150	144	138	132	126	
	87005f	875f5f	87875f	87af5f	87d75f	87ff5f	afff5f	afd75f	afaf5f	af875f	af5f5f	af005f
89	95	101	107	113	119	155	149	143	137	131	125	
	870000	875f00	878700	87af00	87d700	87ff00	afff00	afd700	afaf00	af8700	af5f00	af0000
88	94	100	106	112	118	154	148	142	136	130	124	
	d70000	d75f00	d78700	d7af00	d7d700	d7ff00	ffff00	ffd700	ffaf00	ff8700	ff5f00	ff0000
160	166	172	178	184	190	226	220	214	208	202	196	
	d7005f	d75f5f	d7875f	d7af5f	d7d75f	d7ff5f	ffff5f	ffd75f	ffaf5f	ff875f	ff5f5f	ff005f
161	167	173	179	185	191	227	221	215	209	203	197	
	d70087	d75f87	d78787	d7af87	d7d787	d7ff87	ffff87	ffd787	ffaf87	ff8787	ff5f87	ff0087
162	168	174	180	186	192	228	222	216	210	204	198	
	d700af	d75faf	d787af	d7afaf	d7d7af	d7ffaf	ffffaf	ffd7af	ffafaf	ff87af	ff5faf	ff00af
163	169	175	181	187	193	229	223	217	211	205	199	
	d700d7	d75fd7	d787d7	d7afd7	d7d7d7	d7ffd7	ffffd7	ffd7d7	ffafd7	ff87d7	ff5fd7	ff00d7
164	170	176	182	188	194	230	224	218	212	206	200	
	d700ff	d75fff	d787ff	d7afff	d7d7ff	d7ffff	ffffff	ffd7ff	ffafff	ff87ff	ff5fff	ff00ff
165	171	177	183	189	195	231	225	219	213	207	201	
	080808	121212	1c1c1c	262626	303030	3a3a3a	444444	4e4e4e	585858	626262	6c6c6c	767676
232	233	234	235	236	237	238	239	240	241	242	243	
	eeeeee	e4e4e4	dadada	d0d0d0	c6c6c6	bcbbbb	b2b2b2	a8a8a8	9e9e9e	949494	8a8a8a	808080
255	254	253	252	251	250	249	248	247	246	245	244	
	000000	cd0000	00cd00	cdcd00	0000ee	cd00cd	00cdcd	e5e5e5				
0	1	2	3	4	5	6	7					
	7f7f7f	ff0000	00ff00	ffff00	5c5cff	ff00ff	00ffff	ffffff				

Посредине — обозначение цвета для цвета курсора, снизу — обозначение цвета для текста. Как вы можете увидеть, что я для текста и курсора использую цвет морской волны. Т.к. название сервера `retrel` («буревестник»), то он ассоциируется у меня с этим цветом.

Теперь `.bashrc`, тоже показываю его не полностью, а только то что имеет отношение к теме:

```
# .bashrc
# local stuff
[[ -f ~/.bash_local ]] && . ~/.bash_local
```

Тут я вставляю код из `.bash_local` в общий файл. Таким образом определяться ранее описанные переменные с цветом сервера.

```
# set to red
root_cursor_color='#FF0000'
root_prompt_color='196'
```

Еще две переменные определяю с чисто красным цветом, он будет использоваться для маркировки терминалов привелигированного пользователя (`root'a`).

```
#my favorite PS1
case "$TERM" in
xterm*|rxvt*)
    PS1='\[\e[38;5;${prompt_color}m\]\t j\j \u@\h:\w\n\''
    [[ $UID == 0 ]] && { prompt_color=$root_prompt_color; cursor_color=$root_cursor_color; }
    PS1="$PS1""'\[\e[m\e]12;${cursor_color}\a\e[38;5;${prompt_color}m\]\$ \[\e[m\]'
    ;;
*)
    PS1='\t j\j \u@\h:\w\n\$\''
    ;;
esac
```

Тут проверяется какой используется терминал. Для любого неизвестного или неподдерживаемого цвета будет использоваться приглашение без цвета (`PS1='\t j\j \u@\h:\w\n\$\'`) так, как я это описал в начале статьи. Но если имя терминала начинается на `xterm` или `rxvt`, например, так себя позиционирует терминал Gnome, начинаем кудесить с цветом. Первая строчка — задаем цвет текста — цвет сервера и выводим первую строку приглашения ввода команд. Она всегда будет окрашена в цвет сервера. Вторая строчка — проверяем, работаем ли мы под непривелигированным или привелигированным пользователем (`root'ом`). Если `root` — то переопределяем цвета на красный. Третья строчка — формируем вторую строку приглашения и определяем цвет курсора в терминале. Т.е. там у нас получится либо `$` и через пробел курсор, оба покрашенные в цвет сервера, если пользователь обычный. Либо красный `#` и через пробел красный курсор, если это `root`.

```
# If this is an xterm set the title to user@host:dir
case "$TERM" in
xterm*|rxvt*)
    PS1="\[\e]0;${debian_chroot:+($debian_chroot)}\u@\h: \w\a\]$PS1"
    ;;
*)
    ;;
esac
```

esac

А это, если честно, один в один скопированно из первоначального .bashrc от Дебиана. Знаю, что этот код видоизменяет title bar у окна, размещает там информацию об пользователе, сервере и домашней директории. Но поскольку этот код придумал не я, комментировать его не буду.

В результате у нас должно получится так, как на картинке в самом начале публикации.

linux bash scripts, linux для всех, linux, terminal emulator, unix, xterm, gnome-terminal



16078 ★ 259



Олег Самойлов @splarv

карма рейтинг
11,0 24,1

Комментарии (42)



Crandel 2 ноября 2015 в 12:52 #

0

Все равно fish намного приятнее bash и в плане подсветки и автодополнения и написания скриптов



splarv 2 ноября 2015 в 14:01 # ↑

0

Дело не в bash. Эти ESC последовательности команд зашиты в PS1 читаются терминалом и терминал, а не bash их расцветчивает. Так что будут работать с любым шелом. И чем еще удобен так то что при этом на сам терминал не завязано. Будешь в одном терминале переходить по ssh с сервера на сервер и цвета у приглашения будут меняться соответственно. Ну и цвет курсора, когда открыты конфиги в редакторе, например, беглово взгляда на курсор достаточно чтобы не забывать — на каком сервере.



Crandel 2 ноября 2015 в 14:40 (комментарий был изменён) # ↑

-6

Ничего подробного не увидел. У меня тоже PS1 в bash расцветчен. переходю с bash по ssh на другой сервер, а там стандартное приветствие. По моему, вы тут какое-то ерунду написали. У меня fish даже в ту работает нормально, без графического терминала. PS1 — bash переменная, и регулирует ее вывод.



splarv 2 ноября 2015 в 14:59 # ↑

+3

Разумеется конфиги (.bashrc или что там у вас) должны быть на каждом сервере, каждый в свой цвет. Сами по себе цвета не появятся. :) Аналог PS1 у fish устроен по другому, там это функция fish_prompt. Но суть та же самая, если будите там выдывать ESC последовательности управляющие цветом — они будут интерпретироваться терминалом. Шелл только принимает символы и выдает их, в том числе ESC последовательности и прочие управляющие символы, а цвета воспроизводит всегда терминал.

Bash тоже в линукс консоле работает. Но цвета линукс консоле цвета по беднее и их по другому надо будет задавать, цвет курсора там задать не получилось и набор цветов там будет упрощенный.



Crandel 2 ноября 2015 в 15:13 # ↑

-3

Ага, теперь понятнее, я думал без конфига на сервере тоже работать будет



dark_ruby 2 ноября 2015 в 13:04 #

0

в плане автодополнения путей вот еще удобная штука github.com/rupa/z



DarkPark 2 ноября 2015 в 14:00 # ↑

0

github.com/wting/autojump



kvaps 2 ноября 2015 в 13:19 #

+1

А так же больше цветов, еще больше!

```
git config --global color.ui true
alias ls='ls --color=auto'
```

```
alias dmesg='dmesg --color=always'
alias grep='grep --color=always'
alias fgrep='fgrep --color=always'
alias egrep='egrep --color=always'
alias gcc='gcc -fdiagnostics-color=always'
alias pacman='pacman --color=always'
```



kvaps 2 ноября 2015 в 13:27 # ↑

+2

А, еще можно **man** раскрасить:

```
man() {
    env LESS_TERMCAP_mb=$'\E[01;31m' \
    LESS_TERMCAP_md=$'\E[01;38;5;74m' \
    LESS_TERMCAP_me=$'\E[0m' \
    LESS_TERMCAP_se=$'\E[0m' \
    LESS_TERMCAP_so=$'\E[38;5;246m' \
    LESS_TERMCAP_ue=$'\E[0m' \
    LESS_TERMCAP_us=$'\E[04;38;5;146m' \
    man "$@"
}
```



uscr 2 ноября 2015 в 14:00 #

+2

Двухстрочное приглашение — очень удобная штука! Можно впахнуть много информации и при этом останется место для длинной команды без переноса. Я пару лет использую трёхстрочное приглашение:

- 1 имя пользователя, hostname, la, текущее время
- 2 текущий каталог
- 3 для ввода команды

Сначала кажется слишком громоздко, но быстро привык и чувствую дискомфорт при использовании однострочных приглашений.

```
[ uscr@workblock ] [ 0.53 0.43 0.28 ] [ 14:00:08 ]
/usr/src/linux-headers-3.19.0-15/kernel/trace
type command here
```



splarv 2 ноября 2015 в 14:04 # ↑

+3

Ну да, у меня практически тоже самое, только без load, одинаково мыслим. :) Но все это, в принципе, в одну строчку помещается, так что двухстрочного для меня достаточно. Да и удобно еще тем что наглядно отделяет вывод одной команды от другой.



Sild 2 ноября 2015 в 14:07 # ↑

0

Во всем этом деле с датой расстраивает то, что время пишется на момент приглашения. Получить бы время выполнения))



splarv 2 ноября 2015 в 14:12 (комментарий был изменён) # ↑

0

Ну так если работаешь непрерывно, то по разнице времени на «момент приглашения» (т.е. на момент завершения задачи) с предыдущим промптом можно приблизительно оценить время выполнения. А для более точной оценки есть команда **time**:

```
14:09:35 j0 olleg@petrel:~
$ time sleep 10

real    0m10.009s
user    0m0.000s
sys     0m0.000s
14:09:51 j0 olleg@petrel:~
$
```



Sild 2 ноября 2015 в 14:20 # ↑

0

Как-то я не подумал про следующее приглашение. Действительно, задачи как правило выполняются меньше

секунды, редко больше 5 секунд. Этой оценки вполне хватит.
Пойду подпиливать .bashrc



SabMakc 2 ноября 2015 в 15:10 # ↑

0

Можно и время выполнения получить:

jakemccrary.com/blog/2015/05/03/put-the-last-commands-run-time-in-your-bash-prompt



Sild 2 ноября 2015 в 15:12 # ↑

0

Извините, я всех запутал с терминологией, меня интересовало именно время начала исполнения. Но длительность выполнения тоже стоит взять на вооружения.



xenohunter 2 ноября 2015 в 14:41 # ↑

0

Спасибо за идею, попробую. И [splarv](#) тоже спасибо за статью!



uscr 2 ноября 2015 в 14:49 # ↑

+2

То, что у меня, делается вот так:

```
PS1="\[\e[0;36m\] ──\[\e[0m\] [ \[\e[0;33m\]\u\[\e[0m\]$txtgrn$txtcyn\h$txttrst ] [$txtylw\$(load_averag)$txt  
rst] [ $txtcyn\t$txttrst ]\n$txtcyn└─ $txtgrn\w$txtcyn\n$txtcynL>$txttrst "
```

Цвета заданы так же в отдельном файлике переменными. \$txttrst отменяет все модификации цвета (txttrst="\[\e[0m\]").
\$(load_averag) — функция. Парсит вывод uptime выдирая LA.



d7s2di 3 ноября 2015 в 10:58 # ↑

0

А по-моему, двухстрочное приглашение — монстроузное приглашение, а вся выведенная информация избыточна и выводится либо средствами оконного менеджера (часы), либо не шибко полезна, чтобы постоянно висеть перед глазами (loadavg).

Для себя я подобрал оптимальную конфигурацию: hostname, текущий каталог с сокращением до половины ширины терминала и информация о свободном месте на текущей файловой системе.



Terranz 3 ноября 2015 в 11:01 (комментарий был изменён) # ↑

+1

а если оконного менеджера нет?

у меня выводится свободная память, время, текущий каталог и количество задач в бекграунде
но это, конечно, вкусовщина



d7s2di 3 ноября 2015 в 11:13 # ↑

0

>а если оконного менеджера нет?

Когда я подключаюсь по ssh, всегда использую tmux или screen (на совсем уж древних системах). Отложенные и возобновленные сессии — это очень удобно, плюс мультиплексор имеет свой персональный статусбар, куда можно вывести часики и имя хоста.

Но если уж вот прям никак: мультиплексор нельзя, оконного менеджера нет, кругом только чОрная консоль, то проще набрать три буквы команды top или четыре команды date, чем постоянно держать перед глазами мусор, который, к слову, не будет автоматически обновляться.

Помнится, раньше было очень модно городить сопку и выводить туда такую «полезную» информацию, как версия ядра, хостнейм своего локалхоста, несколько штук часов, календарик... Это вот все напоминает.



splarv 3 ноября 2015 в 11:42 (комментарий был изменён) # ↑

+1

Даже такой минимум как хостнейм и текущий каталог (а это по умолчанию) порой занимают порядочную ширину и для набора команд остается недостаточно места. Да и неудобно это. Гораздо приятнее набирать команды с абсолютно пустой строки. :) Это и было основной причиной двухстрочного приглашения. Плюс оно удобно отделяет вывод предыдущей команды. А вывод времени в основном используется не для того чтобы знать текущее время, а для того чтобы понимать насколько какая программа «подвисла», сколько выполняется длительная компиляция и т.д. Я понимаю что time удобнее. Но подобный интерес зачастую возникает не заранее, а после того как программа неожиданно долго работает. :) Так что у меня почти тоже что и у вас, разве что вывода свободного места нет, т.к.

набрать df всегда просто.



d7s2di 3 ноября 2015 в 12:23 # ↑

-1

>Даже такой минимум как жостеям и текущий каталог (а это по умолчанию) порой занимает порядочную ширину

Потому, я обычно делаю вот так:

```
local DPRINTDN
{{ DPRINTDN = ${COLUMNS} / 3 }}
local CUR="[ %DPRINTDN<...<8-4< ]"
PROMPT="%$CUR% ${COLUMNS}CUR$DEFAULT ->"
```

Путь срежется до трети ширины терминала. Вот двустороннее приглашение я люблю: справа у меня и выводится

```
df -hP . | sed -n '2p' | awk -F ' ' '{print $4}'
```

Вообще, люблю подобные обсуждения: всегда можно посмотреть что-то полезное или поделиться своим.



chill84 2 ноября 2015 в 15:22 #

+1

[а еще можно эмодзи добавить](#)

удобно, чтобы отличать один проект от другого



splarv 2 ноября 2015 в 15:32 # ↑

0

А вот это интересно. Про вывод картинок в терминале я не слышал. :)



Ascott 2 ноября 2015 в 20:19 # ↑

0

Потому что это не картинки, а unicode символы. Другое дело, нужно, что бы баш их умел правильно «рисовать».



splarv 3 ноября 2015 в 11:52 # ↑

+1

Не баш, а терминал. Должна быть или специальная поддержка со стороны терминала, чтобы он вставлял туда картинки. Либо их должен поддерживать системный фонт. Оказалось что в dejavu есть много смайликов и даже смайлики-котята. Можно использовать в терминале. :)



sav13 2 ноября 2015 в 15:50 #

0

Да.

А во времена UNIXов, когда еще на было BASH и LINUX украшали экран ESC-последовательностями



Terranz 2 ноября 2015 в 17:22 (комментарий был изменён) #

+3

больше десяти лет назад в хакере нашёл себе хороший, годный вариант оформления командной строки

```
get_freemem ()
{
echo -n `free | grep Mem | awk '{print $4}'`
}
```

```
export PS1 forem="(w)=(l)=(`get_freemem`)=(j:~$?)=\\n=>"
```

<https://habrastorage.org/files/5b1/973/fb9/5b1973fb906a459492e70cab0c397b97.png>



Meklon 2 ноября 2015 в 18:13 # ↑

+3

Проиллюстрирую, раз вы с отрицательной кармой:

```
= (~ / projects) = (18:23:42) = (2189804) = (0:0) =
=> ls -la
```




Gendalph 2 ноября 2015 в 18:42 #


0

```
$(ERR="$?"; if [[ "$ERR" != "0" ]]; then printf "\033[01;37m(%.4s)\033[00m " $ERR $ERR; fi) $PS1
```

добавляет код ошибки, белым в скобках, в начале PS1

 **kt97679** 2 ноября 2015 в 19:43 # 0
Я сторонник минимализма в отношении подсказки, но у меня сохраняется расширенная история. Вот фрагмент .bashrc:

```
HISTTIMEFORMAT='%t%F %T%'
echo $PROMPT_COMMAND|grep -q bash_eternal_history || export
PROMPT_COMMAND="${PROMPT_COMMAND:+$PROMPT_COMMAND; } history -a; "echo -e ${USER}@$HOSTNAME
\\$PWD\\t"$(history 1)" >> ~/.bash_eternal_history'
```


 **felicast** 2 ноября 2015 в 20:39 # 0
Добавлю свои 5 копеек. Добавляет текущую ветку для репозитория. Для git и hg репозиториях.
image


```
hg_branch() {
    BRANCH=`hg branch 2> /dev/null | awk '{print "(hg:"$1")"}'`
    if [[ $BRANCH == *release* ]]
    then
        BRANCH="\e[31m$BRANCH"
    fi
    if [[ $BRANCH == *default* ]]
    then
        BRANCH="\e[31m$BRANCH"
    fi
    echo -e $BRANCH
}
```


в PS1:


```
$(__git_ps1 "(git:%s)") $(hg_branch)
```


ps: условия в hg_branch() раскрашивают блок в красный цвет, чтобы видно было, что не надо ничего коммитить в default и release

 **xvilka** 3 ноября 2015 в 10:24 # 0
256 цветов в терминале — прошлый век. Большинство современных терминалов имеют поддержку 16 миллионов цветов (True Color). Вот [здесь](#) я собрал список терминалов, которые их поддерживают, и не поддерживают. Для большинства тех, которые не поддерживают — есть сторонние патчи или форки. Ведется работа по интеграции их в мейнстрим. Есть даже Pull Request в tmux github.com/tmux/tmux/pull/112

 **Meklon** 3 ноября 2015 в 10:47 # ↑ +1
Вроде оно логично, но нафига вам оттенки цвета «бедра испуганной нимфы»? По большому счету тут будут использоваться только чистые сатурированные тона, как правило. Красный, синий, желтый, зеленый... Человеческий мозг не в состоянии удерживать даже 256 категорий одновременно. Что там может быть? Продакшен, тестовый нестабильный, тестовый предпродакшен и т.п. Ну 10 категорий. Дальше смысл теряется, на мой взгляд.

 **splarv** 3 ноября 2015 в 11:49 # ↑ 0
Вот это интересно, если оно так. У вас значится что Gnome Terminal тоже поддерживает. Напишите пример с echo который бы позволил задать цвет текста в 24хбитном RGB. Ваш пример по ссылке пробовал — не работает.

 **xvilka** 3 ноября 2015 в 13:01 # ↑ 0
В gist-е упомянуто, что это применимо только к Gnome Terminal скопированным Gtk+3, и libvte старше 0.36 версии.

 **kstep** 3 ноября 2015 в 11:54 (комментарий был изменён) # +2
Раз пошла такая пьянка, моя строка выглядит так:

```

- ~/git/pure master?! pts/7: sleep 5
^C
- ~/git/pure master?! pts/7<130>: sleep 6
- ~/git/pure master?! pts/7 6s: blah
zsh: command not found: blah
- ~/git/pure master?! pts/7<127>:

```

Доступно в моём zsh-config репе: [prompt.zsh](#)

Выводит:

- 1) текущий каталог, если слишком длинный, то последние 5 каталогов,
- 2) текущая git-ветка, если я в репозитории (иначе не выводит этот компонент), вопросительный знак если есть незакоммиченные изменения, восклицательный — если есть изменения в стейджинге,
- 3) текущий терминал (удобно различать в каком терминале находишься и какой шелл убивать если что),
- 4) если команда выполнялась дольше 5 секунд, выводит сколько времени она выполнялась,
- 5) если команда неудачно завершилась, то выводит код ошибки.

Итого всё очень минималистично, выводит только ту информацию, которая важна по контексту, что позволяет держать строку очень короткой, но при этом информативной.

И да, для bash придётся переписать, но тут главное идея.



Meklon 3 ноября 2015 в 11:55 #

+2

Вообще, мне больше нравится разыернутый синтаксис. Ненавижу перлообразные нагромождения, для меня они нечитаемы. Пример из Арчевики:

```

set_prompt () {
    Last_Command=$? # Must come first!
    Blue='\[\e[01;34m\]'
    White='\[\e[01;37m\]'
    Red='\[\e[01;31m\]'
    Green='\[\e[01;32m\]'
    Reset='\[\e[00m\]'
    FancyX='\342\234\227'
    Checkmark='\342\234\223'

    # Add a bright white exit status for the last command
    PS1="$White$? "
    # If it was successful, print a green check mark. Otherwise, print
    # a red X.
    if [[ $Last_Command == 0 ]]; then
        PS1+="$Green$Checkmark "
    else
        PS1+="$Red$FancyX "
    fi
    # If root, just print the host in red. Otherwise, print the current user
    # and host in green.
    if [[ $EUID == 0 ]]; then
        PS1+="$Red\\h "
    else
        PS1+="$Green\\u@\\h "
    fi
    # Print the working directory and prompt marker in blue, and reset
    # the text color to the default.
    PS1+="$Blue\\w \\\$Reset "
}
PROMPT_COMMAND='set_prompt'

```



splarv 3 ноября 2015 в 14:36 # ↑

0

Да я тоже за развернутый синтаксис. :) Но создать переменные для всех 255 цветов... (причем и для обозначений цвета для букв и для курсора, т.е. 510). Там даже уникальные имена цветам замучаешься придумывать, а когда придумаешь — забудешь какой конкретный цвет они обозначают. :)



Meklon 3 ноября 2015 в 15:47 # ↑

0


А нафига вам 256 цветов? Я вообще смысла не вижу больше 10-15 использовать. Как писал уже выше, вы категории не запомните. Ну не может человеческий мозг разбивать множество на 100-200 категорий. Сводим к 3-5 обычно.

 **splarv** 3 ноября 2015 в 16:48 # ↑ +1


Вы правы, я использую ровно 5. Но выбирал то я тх из 255. :) Я за свободу выбора.

Только зарегистрированные пользователи могут оставлять комментарии. [Войдите](#), пожалуйста.

Что обсуждают


Сейчас	Вчера	Неделя	Месяц	
				
Вышел JetBrains Toolbox со всеми обновленными десктопными продуктами				129
Geek Week 2015: учиться, учиться и ещё раз учиться				13
Покончите с беспределом: внедрите бизнес-процессы в CRM				28
Защита in-App Purchase iOS от ломалок с помощью сервера				4
Do good code: 8 правил хорошего кода				52


Самое читаемое


Сейчас	Вчера	Неделя	Месяц	
				
10 относительно честных способов взломать почту				15k
БЭМ — методология развешивания костылей				4k
Технические собеседования: советы				7k
Ошибки и проблемы серверов большой тройки: часть первая. Dell				2k
Эксплоит на миллион. У нас есть победитель				8k


Лучшее на Geektimes


- 1

Разработка power bank для ноутбука. От макета к готовому изделию. Часть первая  17
- 2

Как перестать бояться ночной темноты и полюбить Вселенную  5
- 3

Как сделать наушники с костной проводимостью звука дома за 5 минут и разбор популярных заблуждений об этой технологии  15
- 4

Microsoft понижает лимиты на OneDrive, а для России повышает цены на ПО и подписки  12
- 5

Какой квадрокоптер мне выбрать, если я хочу...? FAQ от Dronk.Ru  15

Все публикации Популярные хабы Компании

Лучшее на Мегамозге

- 1 [Иностранные IT-компании могут уйти из РФ: читаем новость по памятке Галины Тимченко](#) 3
- 2 [Продвижение мобильных приложений: выученные уроки](#) 1
- 3 [Блог Пола Грэма: Как распознать предвзятость?](#)
- 4 [Ритейл уходит в небо: дроны-курьеры](#) 1
- 5 [12 топовых финтех компаний родом из Скандинавии](#) 5

Все публикации Популярные хабы Стартапы

Вакансии на «Моём круге»

Frontend Developer

Санкт-Петербург • Полный рабочий день

Senior Java developer

Санкт-Петербург • Полный рабочий день

Разработчик приложений для iOS

Москва • Полный рабочий день

Дизайнер интерфейсов (мобильные приложения)

Москва • Полный рабочий день

Фронтэнд разработчик / Frontend JS Developer

Москва • Полный рабочий день

Lead / Senior Python Developer

Москва • Полный рабочий день

C#/XAML Разработчик

Москва • Полный рабочий день

Разработчик (программист) ASP.NET MVC

Санкт-Петербург • Полный рабочий день

Старший разработчик C++

Санкт-Петербург • Полный рабочий день

Стажёр-тестировщик (Junior QA Engineer)

Минск • Полный рабочий день

разместить вакансию

все вакансии

Заказы на «Фрилансим»

Помочь в портировании Qt приложения в Arm Embedded system

03.11.2015 • 0 откликов

Верстка макетов сайта

03.11.2015 • 5 откликов

Задачи по существующему проекту на MeteorJS

03.11.2015 • 0 откликов

разместить заказ

все заказы

Иллюстрация для термокружки

03.11.2015 • 5 откликов

BackEnd для соц сети путешественников

03.11.2015 • 11 откликов

Иллюстрация для календаря

03.11.2015 • 11 откликов

Настроить pptp через Cisco ASA 5505

03.11.2015 • 3 отклика

Аплоадер объявлений в тизерные сети

31.10.2015 • 4 отклика

(ДО!)Работка корпоративного портала на MOD-X Revo

03.11.2015 • 3 отклика

Подготовить отчетные документы по проекту

03.11.2015 • 3 отклика