How to Change Bash Prompt

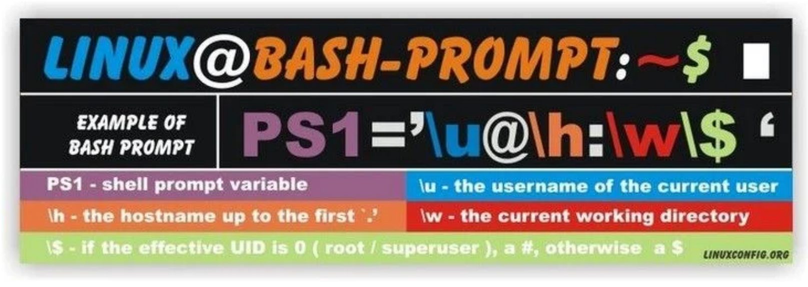
Lubos Rendek Programming & Scripting
02 June 2020

Contents

Default bash command line prompt on many Linux systems is quite minimal. As we will see in this article, it can be easily changed by modifying bash `PS{n}` variables, so to include information such as display time, load, number of users using the system, uptime and more.

In this tutorial you will learn:

- What are PS1 and PS2 shell variables
- How to create custom shell prompts
- What are the characters we can use to customize a shell prompt


LINUX@BASH-PROMPT:~$
EXAMPLE OF BASH PROMPT
PS1='\u@\h:\w\$ '
PS1 - shell prompt variable
\u - the username of the current user
\h - the hostname up to the first '.'
\w - the current working directory
\$ - if the effective UID is 0 (root / superuser), a #, otherwise a $
LINUXCONFIG.ORG

(/images/linux-bash-prompt.jpg)

Linux bash prompt

Software Requirements and Conventions Used

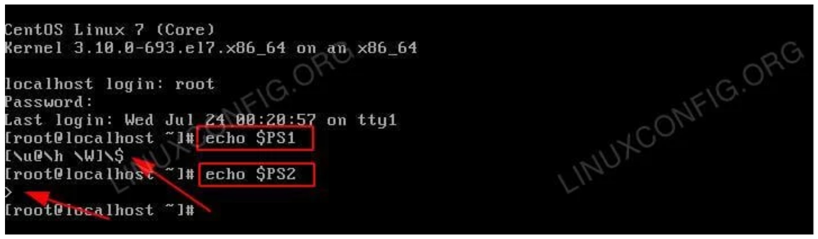
Software Requirements and Linux Command Line Conventions

Category	Requirements, Conventions or Software Version Used
System	Distribution-independent
Software	No special software is needed to follow this tutorial
Other	Minimal knowledge of the Bash shell
Conventions	**#** - requires given linux commands (/linux-commands) to be executed with root privileges either directly as a root user or by use of `sudo` command **$** - requires given linux commands (/linux-commands) to be executed as a regular non-privileged user

Bash prompt variables

As anything else in the Linux system also bash prompt can be customized. We can accomplish the task by changing the values of bash `PS1`, `PS2`, `PS3`, `PS4` variables. To keep the things simple, this article will be concerned just with the first two. Use echo command to see their values:

```
$ echo "Bash PS1 variable:"  $PS1
$ echo "Bash PS2 variable:"  $PS2
```



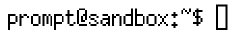
```
CentOS Linux 7 (Core)
Kernel 3.10.0-693.el7.x86_64 on an x86_64

localhost login: root
Password:
Last login: Wed Jul 24 00:20:57 on tty1
[root@localhost ~]# echo $PS1
[\u@\h \W]\$
[root@localhost ~]# echo $PS2
>
[root@localhost ~]#
```

(/images/Ps1_ps2_bash_prompt.jpg)

PS1 and PS2 bash prompt

Bash PS1 prompt variable

`PS1` is a primary prompt variable. Currently it holds `\u@\h:\w\$` special bash characters. This is the default structure of the bash prompt on many Linux systems and is displayed every time you log in using a terminal. Please see the following section "Bash prompt special characters" for explanation of `\u`, `\h`, `\w` and `\$` symbols. Here is a classical bash prompt with default settings:



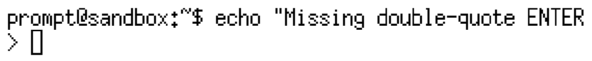
```
prompt@sandbox:~$ ▯
```

(/images/Ps1_bash_prompt.gif)

PS1 bash prompt

Bash PS2 prompt variable

`PS2` bash shell variable is a secondary prompt. This prompt is displayed if the shell waits for a user input, for example you forget to insert second quotation.



```
prompt@sandbox:~$ echo "Missing double-quote ENTER
> ▯
```

(/images/Ps2_bash_prompt.gif)

PS2 bash prompt

Bash prompt special characters

Bash prompt can be customized by using special characters. Here is a quick overview of the most used characters and their meaning:

Bash prompt special characters

Bash special character	Bash special character explanation	Bash special character	Bash special character explanation
\\a	an ASCII bell character (07)	\\d	the date in "Weekday Month Date" format (e.g., "Tue May 26")
\\]	end a sequence of non-printing characters	\\e	an ASCII escape character (033)
\\h	the hostname up to the first `.'	\\H	the hostname
\\j	the number of jobs currently managed by the shell	\\l	the basename of the shell's terminal device name
\\n	newline	\\r	carriage return
\\s	the name of the shell, the basename of $0 (the portion following the final slash)	\\t	the current time in 24-hour HH:MM:SS format
\\T	the current time in 12-hour HH:MM:SS format	\\@	the current time in 12-hour am/pm format

Bash special character	**Bash special character explanation**	**Bash special character**	**Bash special character explanation**
\\A	the current time in 24-hour HH:MM format	\\u	the username of the current user
\\v	the version of bash (e.g., 2.00)	\\V	the release of bash, version + patchelvel (e.g., 2.00.0)
\\w	the current working directory	\\W	the basename of the current working directory
\\!	the history number of this command	\\#	the command number of this command
\\$	if the effective UID is 0, a #, otherwise a $	\\nnn	the character corresponding to the octal number nnn
\\\\	a backslash	\\[	begin a sequence of non-printing characters, which could be used to embed a terminal control sequence into the prompt
\\D{format}	the format is passed to strftime(3) and the result is inserted into the prompt string; an empty format results in a locale-specific time representation. The braces are required		

Bash prompt customization

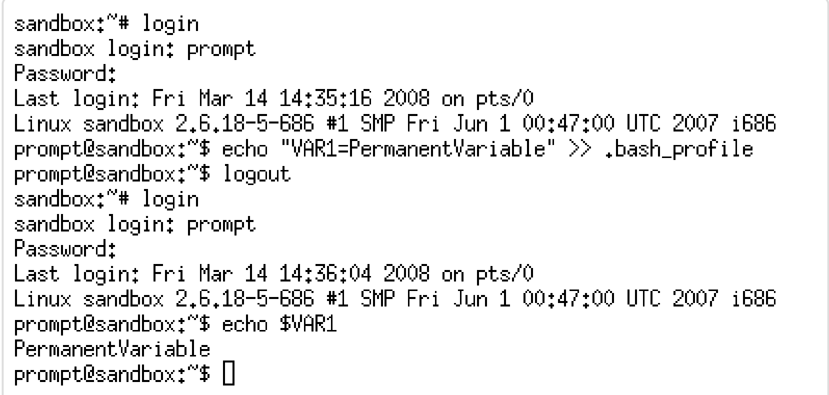
After user logins into the system, user environment variables are initialized from various files:

- /etc/profile or /etc/bashrc (system wide)
- ~/.bash_profile , ~/.bash_login , ~/.profile , ~/.bashrc or ~/.bash_logout (user)

It is important to know that all users environment variable have a life time equal to the terminal session. When the terminal session is closed the user's variables including bash shell variables (/bash-scripting-tutorial-for-beginners#h11-variables) defined during a terminal session are emptied and a again redefined when new terminal session is created either via logo in shell or interactive shell. Lets define two variables to prove this statement.

Permanent bash variable definition

First, we will define a permanent variable in one of the bash initialization files, `~/.bash_profile`, then we will define a temporary variable in the shell prompt. Let's define permanent user variable:



```
sandbox:~# login
sandbox login: prompt
Password:
Last login: Fri Mar 14 14:35:16 2008 on pts/0
Linux sandbox 2.6.18-5-686 #1 SMP Fri Jun 1 00:47:00 UTC 2007 i686
prompt@sandbox:~$ echo "VAR1=PermanentVariable" >> .bash_profile
prompt@sandbox:~$ logout
sandbox:~# login
sandbox login: prompt
Password:
Last login: Fri Mar 14 14:36:04 2008 on pts/0
Linux sandbox 2.6.18-5-686 #1 SMP Fri Jun 1 00:47:00 UTC 2007 i686
prompt@sandbox:~$ echo $VAR1
PermanentVariable
prompt@sandbox:~$
```

(/images/Bash_permanent_variable.gif)

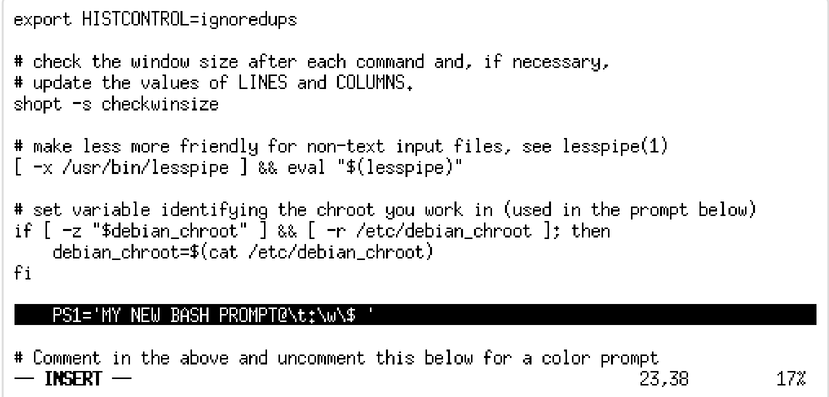
Bash permanent variable

What happened here, is that user "prompt" modified its own .bash_profile initialization file located in his/her home directory by appending a `VAR1` variable definition. When user "prompt" logged out and logged in again the `$var1` variable is initialized and available for the new terminal session.

On the same principles we can define our bash prompt. The best place to do it is that bash initialization file `.~/bashrc`. Open up your `~/.bashrc` file and add/edit the line defining a `PS1` variable to something like:

```
PS1='MY NEW BASH PROMPT@\t:\w\$ '
```

NOTE: Your ~/.barshrc file may differ from the example below !



```
export HISTCONTROL=ignoredups

# check the window size after each command and, if necessary,
# update the values of LINES and COLUMNS.
shopt -s checkwinsize

# make less more friendly for non-text input files, see lesspipe(1)
[ -x /usr/bin/lesspipe ] && eval "$(lesspipe)"

# set variable identifying the chroot you work in (used in the prompt below)
if [ -z "$debian_chroot" ] && [ -r /etc/debian_chroot ]; then
    debian_chroot=$(cat /etc/debian_chroot)
fi

    PS1='MY NEW BASH PROMPT@\t:\w\$ '

# Comment in the above and uncomment this below for a color prompt
-- INSERT --                                          23,38         17%
```

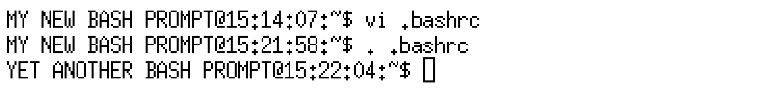
(/images/Bash_new_prompt_define.gif)

New prompt

```
$ source .bashrc
```

or similarly:

```
$ . .bashrc
```



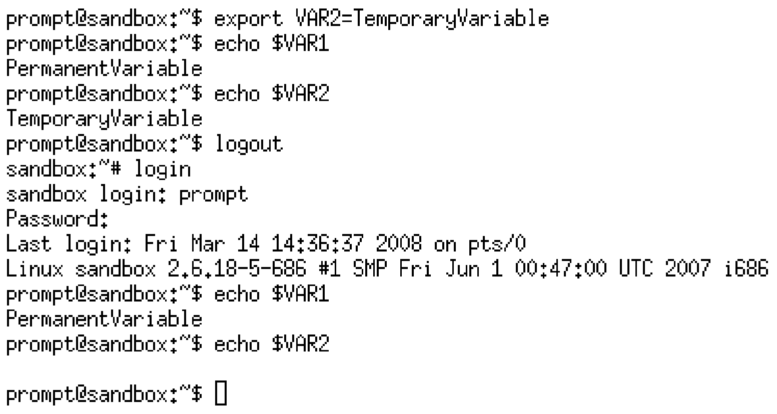
```
MY NEW BASH PROMPT@15:14:07:~$ vi .bashrc
MY NEW BASH PROMPT@15:21:58:~$ . .bashrc
YET ANOTHER BASH PROMPT@15:22:04:~$ 
```

(/images/Bash_another_prompt_define.gif)

Another prompt

Temporary bash variable definition

A temporary bash variable lasts only as long as the current terminal session. This is tome by an export command.



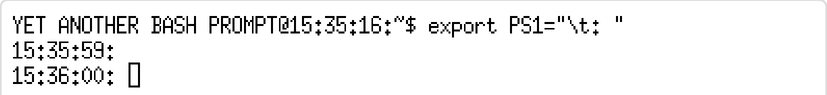
```
prompt@sandbox:~$ export VAR2=TemporaryVariable
prompt@sandbox:~$ echo $VAR1
PermanentVariable
prompt@sandbox:~$ echo $VAR2
TemporaryVariable
prompt@sandbox:~$ logout
sandbox:~# login
sandbox login: prompt
Password:
Last login: Fri Mar 14 14:36:37 2008 on pts/0
Linux sandbox 2.6.18-5-686 #1 SMP Fri Jun 1 00:47:00 UTC 2007 i686
prompt@sandbox:~$ echo $VAR1
PermanentVariable
prompt@sandbox:~$ echo $VAR2

prompt@sandbox:~$ 
```

(/images/Bash_temporary_variable.gif)

Bash temporary variable

As you can see the variable `$VAR2` is not defined when user closes his/her terminal session. The permanent variable `$VAR1` is always defined from the bash initialization file: `~/.bash_profile`. As we can use an export command to define new bash variables we can also use it to modify a bash prompt `$PS1` variable. To change a current bash prompt to display only time we could do:

```
export PS1="\t: "
```



```
YET ANOTHER BASH PROMPT@15:35:16:~$ export PS1="\t: "
15:35:59:
15:36:00: 
```

(/images/Bash_temporary_prompt_define.gif)

Temporary prompt definition

US21PM

Ad Uber Eats

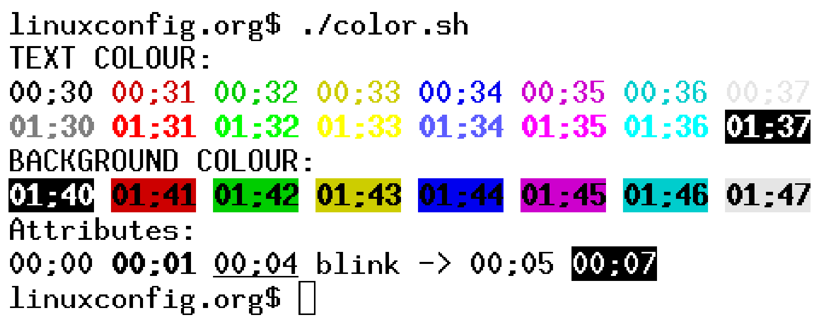
Learn more

Changing foreground and background bash prompt colors

Syntax for changing colors in the bash is as follows:

```
\033[ - Indicates the beginning of color in the
text
x;yzm - Indicates color code
\033[00m - Indicates the end of color in the
text
```

Bash color codes:

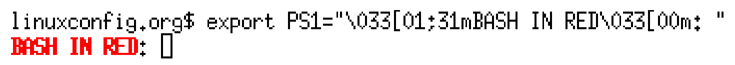


```
linuxconfig.org$ ./color.sh
TEXT COLOUR:
00;30 00;31 00;32 00;33 00;34 00;35 00;36 00;37
01;30 01;31 01;32 01;33 01;34 01;35 01;36 01;37
BACKGROUND COLOUR:
01;40 01;41 01;42 01;43 01;44 01;45 01;46 01;47
Attributes:
00;00 00;01 00;04 blink -> 00;05 00;07
linuxconfig.org$ 
```

(/images/bash-color-codes.gif)

Bash color codes

```
export PS1="\033[01;31mBASH IN RED\033[00m: "
```


```
linuxconfig.org$ export PS1="\033[01;31mBASH IN RED\033[00m: "
BASH IN RED: 
```
(/images/Bash_prompt_red_color.gif)

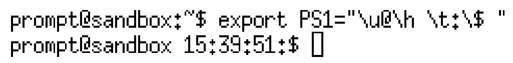
Bash prompt - red color

Bash Prompt Examples

To get you started with your new bash prompt here are couple examples:

Display current Time

```
export PS1="\u@\h \t:\$ "
```

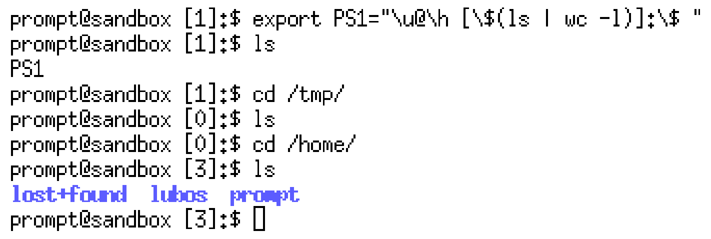

```
prompt@sandbox:~$ export PS1="\u@\h \t:\$ "
prompt@sandbox 15:39:51:$ 
```
(/images/Bash_prompt_current_time.gif)

Bash prompt with current time

Counting Files in the Current Directory

This bash prompt displays current number of files and directories in the current directory.

```
export PS1="\u@\h [\$(ls | wc -l)]:\$ "
```


```
prompt@sandbox [1]:$ export PS1="\u@\h [\$(ls | wc -l)]:\$ "
prompt@sandbox [1]:$ ls
PS1
prompt@sandbox [1]:$ cd /tmp/
prompt@sandbox [0]:$ ls
prompt@sandbox [0]:$ cd /home/
prompt@sandbox [3]:$ ls
lost+found  lubos  prompt
prompt@sandbox [3]:$ 
```
(/images/Bash_prompt_count_files_time.gif)

Bash prompt with files count

Prev (/how-to-perform-web-requests-with-php-using-the-curl-extension)

Next (/how-to-use-graphical-widgets-in-bash-scripts-with-zenity)

MORE ON LINUXCONFIG.ORG:

- Linux Complex Bash One-Liner Examples (/linux-complex-bash-one-liner-examples)
- Bash Scripting Tutorial for Beginners (/bash-scripting-tutorial-for-beginners)
- Useful Bash Command Line Tips and Tricks Examples - Part 6 (/useful-bash-command-line-tips-and-tricks-examples-part-6)
- Bash command line tips and tricks (/bash-command-line-tips-and-tricks)

YOU MAY ALSO BE INTERESTED IN:

Data Visualization - Analytics as a Service

Ad benzait.com

How to Set A Custom Message of The Day on Linux

linuxconfig.org

The 90s Raw And Unedited

Ad History Daily

Bash sc Tutorial

linuxconfig.or

Over 40 and Single?

Ad Brow seSingles.com

How to Debug Bash Scripts

linuxconfig.org

Download file from URL on Linux using command line

linuxconfig.org

Prograr

linuxconfig.or

Comments and Discussions

LINUXCONFIG.ORG FORUM

Start Discussion 0 replies

NEWSLETTER

Subscribe to **Linux Career Newsletter** to receive latest news, jobs, career advice and featured configuration tutorials.

Full Name

Email

☐ **GDPR permission**: I give my consent to be in touch with me via email using the information I have provided in this form for the purpose of news and updates.

☐ I'm not a robot

reCAPTCHA

Privacy - Terms

Subscribe to Newsletter

WRITE FOR US

LinuxConfig is looking for a technical writer(s) geared towards GNU/Linux and FLOSS technologies. Your articles will feature various GNU/Linux configuration tutorials and FLOSS technologies used in combination with GNU/Linux operating system.

When writing your articles you will be expected to be able to keep up with a technological advancement regarding the above mentioned technical area of expertise. You will work independently and be able to produce at minimum 2 technical articles a month.

APPLY NOW (https://www.linuxcareers.com/jobs/floss-technical-writer-new-york-city-new-york/1-1/)

CONTACT

web (at) linuxconfig (dot) org

FEATURED LINUX TUTORIALS

- How To enable the EPEL Repository on RHEL 8 / CentOS 8 Linux (/redhat-8-epel-install-guide)
- Bash scripting Tutorial (/bash-scripting-tutorial)
- How to install VMware Tools on RHEL 8 / CentOS 8 (/how-to-install-vmware-tools-on-rhel-8-centos-8)
- Howto mount USB drive in Linux (/howto-mount-usb-drive-in-linux)
- How to install the NVIDIA drivers on Ubuntu 18.04 Bionic Beaver Linux (/how-to-install-the-nvidia-drivers-on-ubuntu-18-04-bionic-beaver-linux)
- How to update Kali Linux (/how-to-update-kali-linux)
- Ubuntu 20.04 Download (/ubuntu-20-04-download)
- How To Upgrade Ubuntu To 20.04 LTS Focal Fossa (/how-to-upgrade-ubuntu-to-20-04-lts-focal-fossa)
- How to install node.js on RHEL 8 / CentOS 8 Linux (/how-to-install-node-js-on-redhat-8-linux)
- How to check CentOS version (/how-to-check-centos-version)
- How to Parse Data From JSON Into Python (/how-to-parse-data-from-json-into-python)
- Check what Debian version you are running on your Linux system (/check-what-debian-version-you-are-running-on-your-linux-system)
- Bash Scripting Tutorial for Beginners (/bash-scripting-tutorial-for-beginners)
- Ubuntu 20.04 Guide (/ubuntu-20-04-guide)
- How to stop/start firewall on RHEL 8 / CentOS 8 (/redhat-8-stop-start-firewall)
- Install gnome on RHEL 8 / CentOS 8 (/install-gnome-on-redhat-8)
- Linux Download (/linux-download)
- How To Upgrade from Ubuntu 18.04 and 19.10 To Ubuntu 20.04 LTS Focal Fossa (/how-to-upgrade-ubuntu-to-20-04-lts-focal-fossa)
- Enable SSH root login on Debian Linux Server (/enable-ssh-root-login-on-debian-linux-server)

LATEST ARTICLES

LATEST ARTICLES

- How to unrar in Ubuntu (/how-to-unrar-in-ubuntu)
- apt update vs apt upgrade (/apt-update-vs-apt-upgrade)
- How to create modify and delete users account on Linux (/how-to-create-modify-and-delete-users-account-on-linux)
- How to launch external processes with Python and the subprocess module (/how-to-launch-external-processes-with-python-and-the-subprocess-module)
- How to Access Manual Pages for Linux Commands (/how-to-access-manual-pages-for-linux-commands)
- Kali Linux vs Parrot (/kali-linux-vs-parrot)
- Parrot Linux download (/parrot-linux-download)
- How to setup Snap package manager on any Linux distro (/how-to-setup-snap-package-manager-on-any-linux-distro)
- CentOS vs Fedora (/centos-vs-fedora)
- Elementary OS Linux download (/elementary-os-linux-download)
- How to rollback pacman updates in Arch Linux (/how-to-rollback-pacman-updates-in-arch-linux)
- How to read and create csv files using Python (/how-to-read-and-create-csv-files-using-python)
- How to generate and manage ssh keys on Linux (/how-to-generate-and-manage-ssh-keys-on-linux)
- How to Correctly Grep for Text in Bash Scripts (/how-to-correctly-grep-for-text-in-bash-scripts)
- Bash Advanced Variable Idioms for Case Sensitivity Management (/bash-advanced-variable-idioms-for-case-sensitivity-management)