**Unix Variable**

Bash

bash is GNU Bourne-Again SHell. Bash is created by the [Free Software Foundation](https://bash.cyberciti.biz/guide/Free_Software_Foundation), Inc. From the man page:

*Bash is an sh-compatible command language interpreter that executes commands read from the standard input or from a file. Bash also incorporates useful features from the Korn and C shells (ksh and csh). Bash is intended to be a conformant implementation of the Shell and Utilities portion of the IEEE POSIX specification (IEEE Standard 1003.1). Bash can be configured to be POSIX conformant by default.*

The bash is default shell on a Linux, macOS and many other Unix-like systems.

Getting help about bash shell

Type the following command at your terminal to read more about bash:

man bash

Variables

There are two types of variable as follows:

1.System Variables

2.User Defined Variables

System Variables

Created and maintained by [Linux](https://bash.cyberciti.biz/guide/Linux) [bash](https://bash.cyberciti.biz/guide/Bash) shell itself. This type of variable (with the exception of auto\_resume and histchars) is defined in CAPITAL LETTERS. You can configure aspects of the shell by modifying system variables such as PS1, PATH, LANG,HISTSIZE,and DISPLAY etc.

**View All System Variables**

To see all system variables, type the following command at a console / terminal:

set

OR

env

OR

printenv

**Sample Outputs from set command:**

BASH=/bin/bash

BASH\_ARGC=()

BASH\_ARGV=()

BASH\_LINENO=()

BASH\_SOURCE=()

BASH\_VERSINFO=([0]="3" [1]="2" [2]="39" [3]="1" [4]="release" [5]="i486-pc-linux-gnu")

BASH\_VERSION='3.2.39(1)-release'

COLORTERM=gnome-terminal

COLUMNS=158

DBUS\_SESSION\_BUS\_ADDRESS=unix:abstract=/tmp/dbus-FSGj0JzI4V,guid=7f59a3dd0813f52d6296ee404a9a68e1

DESKTOP\_SESSION=gnome

DIRSTACK=()

DISPLAY=:0.0

EUID=1000

GDMSESSION=gnome

GDM\_LANG=en\_IN

GDM\_XSERVER\_LOCATION=local

GNOME\_DESKTOP\_SESSION\_ID=this-is-deprecated

GPG\_AGENT\_INFO=/tmp/gpg-X7NqIv/S.gpg-agent:7340:1

GROUPS=()

GTK\_RC\_FILES=/etc/gtk/gtkrc:/home/vivek/.gtkrc-1.2-gnome2

HISTFILE=/home/vivek/.bash\_history

HISTFILESIZE=500

HISTSIZE=500

HOME=/home/vivek

HOSTNAME=vivek-desktop

HOSTTYPE=i486

IFS=$' \t\n'

LANG=en\_IN

LINES=57

LOGNAME=vivek

MACHTYPE=i486-pc-linux-gnu

MAILCHECK=60

OLDPWD=/home/vivek

OPTERR=1

OPTIND=1

ORBIT\_SOCKETDIR=/tmp/orbit-vivek

OSTYPE=linux-gnu

PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games

PIPESTATUS=([0]="0")

PPID=7542

PS1='${debian\_chroot:+($debian\_chroot)}\u@\h:\w\$ '

PS2='> '

PS4='+ '

PWD=/tmp

SESSION\_MANAGER=local/vivek-desktop:/tmp/.ICE-unix/7194

SHELL=/bin/bash

SHELLOPTS=braceexpand:emacs:hashall:histexpand:history:interactive-comments:monitor

SHLVL=1

SSH\_AGENT\_PID=7339

SSH\_AUTH\_SOCK=/tmp/ssh-IoFXYh7194/agent.7194

TERM=xterm

UID=1000

USER=vivek

USERNAME=vivek

WINDOWID=18874428

WINDOWPATH=7

XAUTHORITY=/home/vivek/.Xauthority

XDG\_DATA\_DIRS=/usr/local/share/:/usr/share/:/usr/share/gdm/

XDG\_SESSION\_COOKIE=186611583e30fed08439ca0047067c9d-1251633372.846960-528440704

\_=set

command\_not\_found\_handle ()

{

if [ -x /usr/lib/command-not-found ]; then

/usr/bin/python /usr/lib/command-not-found -- $1;

return $?;

else

return 127;

fi

}

mp3 ()

{

local o=$IFS;

IFS=$(echo -en "\n\b");

/usr/bin/beep-media-player "$(cat $@)" & IFS=o

}

genpasswd ()

{

local l=$1;

[ "$l" == "" ] && l=16;

tr -dc A-Za-z0-9\_ < /dev/urandom | head -c ${l} | xargs

}

xrpm ()

{

[ "$1" != "" ] && ( rpm2cpio "$1" | cpio -idmv )

}

**Commonly Used Shell Variables**

The following variables are set by the shell:

|  |  |  |
| --- | --- | --- |
| **System Variable** | **Meaning** | **To View Variable Value Type** |
| BASH\_VERSION | Holds the version of this instance of bash. | echo $BASH\_VERSION |
| HOSTNAME | The name of the your computer. | echo $HOSTNAME |
| CDPATH | The search path for the cd command. | echo $CDPATH |
| HISTFILE | The name of the file in which command history is saved. | echo $HISTFILE |
| HISTFILESIZE | The maximum number of lines contained in the history file. | echo $HISTFILESIZE |
| HISTSIZE | The number of commands to remember in the command history. The default value is 500. | echo $HISTSIZE |
| HOME | The home directory of the current user. | echo $HOME |
| IFS | The Internal Field Separator that is used for word splitting after expansion and to split lines into words with the read builtin command. The default value is <space><tab><newline>. | echo $IFS |
| LANG | Used to determine the locale category for any category not specifically selected with a variable starting with LC\_. | echo $LANG |
| PATH | The search path for commands. It is a colon-separated list of directories in which the shell looks for commands. | echo $PATH |
| PS1 | Your prompt settings. | echo $PS1 |
| TMOUT | The default timeout for the read builtin command. Also in an interactive shell, the value is interpreted as the number of seconds to wait for input after issuing the command. If not input provided it will logout user. | echo $TMOUT |
| TERM | Your login terminal type. | echo $TERM export TERM=vt100 |
| SHELL | Set path to login shell. | echo $SHELL |
| DISPLAY | Set X display name | echo $DISPLAY export DISPLAY=:0.1 |
| EDITOR | Set name of default text editor. | export EDITOR=/usr/bin/vim |

* Note you may add above variable (export command) to the initialization file located in the home directory of your account such as ~/[.bash\_profile](https://bash.cyberciti.biz/guide/.bash_profile).

**How Do I Display The Value Of a Variable?**

Use echo command to display variable value. To display the program search path, type:

echo "$PATH"

To display your prompt setting, type:

echo "$PS1"

All variable names must be prefixed with $ symbol, and the entire construct should be enclosed in quotes. Try the following example to display the value of a variable without using $ prefix:

echo "HOME"

To display the value of a variable with echo $HOME:

echo "$HOME"

You must use $ followed by variable name to print a variable's contents.

The variable name may also be enclosed in braces:

echo "**${**HOME**}**"

This is useful when the variable name is followed by a character that could be part of a variable name:

echo "**${**HOME**}**work"

**Say hello to printf**

The [printf command](https://bash.cyberciti.biz/guide/Printf_command" \o "Printf command) is just like [echo command](https://bash.cyberciti.biz/guide/Echo_command) and is available under various versions of UNIX operating systems. It is a good idea to use printf if portability is a major concern for you. The syntax is as follows:

printf "$VARIABLE\_NAME\n"

printf "String %s" $VARIABLE\_NAME

printf "Signed Decimal Number %d" $VARIABLE\_NAME

printf "Floating Point Number %f" $VARIABLE\_NAME

To display the program search path, type:

printf "$PATH\n"

OR

printf "The path is set to %s\n" $PATH

Sample outputs:

The path is set to /home/vivek/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games

Creating and setting variables within a script is fairly simple. Use the following syntax:

varName=someValue

**someValue** is assigned to given **varName** and **someValue** must be on right side of = (equal) sign. If **someValue** is not given, the variable is assigned the null string.

How Do I Display The Variable Value?

You can display the value of a variable with **echo $varName** or **echo ${varName}**:

echo "$varName"

OR

echo "**${**varName**}**"

OR

printf "**${**varName**}**"

OR

printf "%s\n" **${**varName**}**

For example, create a variable called vech, and give it a value 'Bus', type the following at a shell prompt:

vech=Bus

Display the value of a variable vech with echo command:

echo "$vech"

OR

echo "**${**vech**}**"

Create a variable called \_jail and give it a value "/httpd.java.jail\_2", type the following at a shell prompt:

\_jail="/httpd.java.jail\_2"

printf "The java jail is located at %s\nStarting chroot()...\n" $\_jail

However,

n=10 *# this is ok*

10=no# Error, NOT Ok, Value must be on right side of = sign.

**Common Examples**

Define your home directory:

myhome="/home/v/vivek"

echo "$myhome"

Set file path:

input="/home/sales/data.txt"

echo "Input file $input"

Store current date (you can store the output of date by running the shell command):

NOW=**$(**date**)**

echo $NOW

Set NAS device backup path:

BACKUP="/nas05"

echo "Backing up files to $BACKUP/$USERNAME"

**More About ${varName} Syntax**

You need to use ${varName} to avoid any kind of ambiguity. For example, try to print "MySHELL=>$SHELLCode<="

echo "MySHELL=>$SHELLCode<="

Sample outputs:

MySHELL=><=

The bash shell would try to look for an variable called SHELLCode instead of $SHELL. To avoid this kind of ambiguity use ${varName} syntax i.e. ${BASH}Code:

echo "MySHELL=>**${**SHELL**}**Code<="

Sample outputs:

MySHELL=>/bin/bashCode<=

Default shell variables value

You can set the default shell variable value using the following syntax. For example, try to display the value of an undefined variable called grandslam:

echo $grandslam

Nothing will be displayed as the variable grandslam was not set in the first place. If $grandslam unset, set name to "Maria Sharapova", enter:

echo **${**grandslam=Maria Sharapova**}**

Sample outputs:

Maria Sharapova

You can also use the following syntax:

echo **${**grandslam**:-**DefaultValueHere**}**

* if $grandslam name is not set use default "Maria Sharapova":

echo **${**grandslam**:-**Maria Sharapova**}**

* if $grandslam unset, set name to default "Maria Sharapova":

echo **${**grandslam:=Maria Sharapova**}**

The := syntax

If the variable is an empty, you can assign a default value. The syntax is:

${var:=defaultValue}

**Example**

Type the following command at a shell prompt:

echo **${**arg:=Foo**}**

bank=HSBC

echo **${**bank:=Citi**}**

unset bank

echo **${**bank:=Citi**}**

In this example, the [function](https://bash.cyberciti.biz/guide/Writing_your_first_shell_function) die assigns a default value if $1 argument is missing:

die(){

local error=**${**1:=Undefined error**}**

echo "$0: $LINE $error"

}

die "File not found"

die

The second die call will produce an error on screen:

bash: $1: cannot assign in this way

Update the die [function](https://bash.cyberciti.biz/guide/Writing_your_first_shell_function) as follows:

die(){

local error=**${**1**:-**Undefined error**}**

echo "$0: $LINE $error"

}

*# call die() with an argument*

die "File not found"

*# call die() without an argument*

die