**Learn Linux Environment Variables Step-By-Step Easy Guide**

[*2017-02-04*](https://likegeeks.com/linux-environment-variables/)[*admin*](https://likegeeks.com/author/admin/) [Comment(0)](https://likegeeks.com/linux-environment-variables/#respond)

In the previous posts, we talked about some of the basic [**Linux commands**](https://likegeeks.com/basic-linux-commands-part2/), today we continue our journey, and we will talk about something very important in Linux which is Linux Environment Variables.

So what are Environment Variables and what is the benefit of knowing them?

Environment Variables are used to store some values which can be used by scripts from the shell. You are not limited to the shell environment variables, you can create your own.

The bash shell has two types of environment variables:

* **Local variables**
* **Global variables**

The local variable is visible in the shell where it was defined only.

The global variable is visible for any running process that runs from the shell.

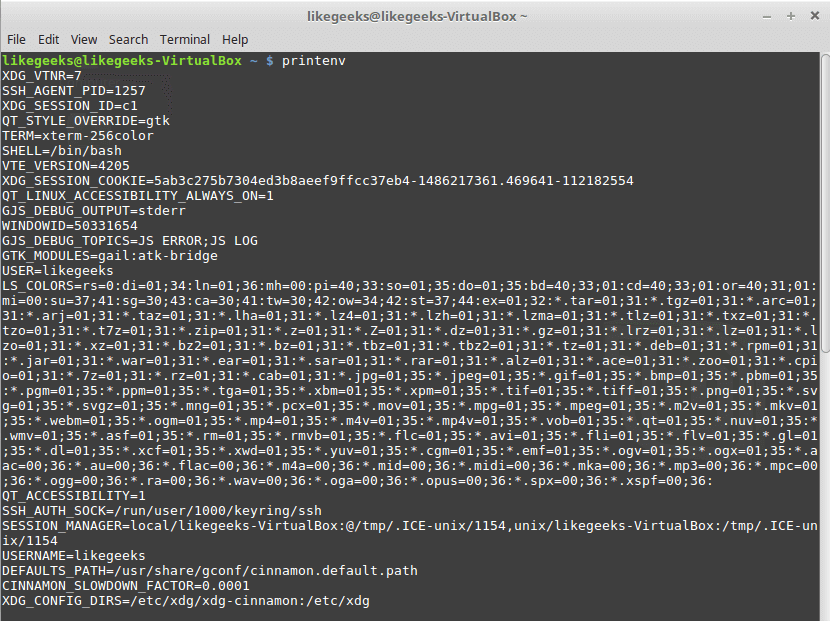
**Table of Contents**[[**hide**](file:///C:\Users\CyberNode\Desktop\unix\Learn%20Linux%20Environment%20Variables%20Step-By-Step%20Easy%20Guide%20-%20Like%20Geeks.htm)][[**hide**](file:///C:\Users\CyberNode\Desktop\unix\Learn%20Linux%20Environment%20Variables%20Step-By-Step%20Easy%20Guide%20-%20Like%20Geeks.htm)]

* [**1 Global variables**](file:///C:\Users\CyberNode\Desktop\unix\Learn%20Linux%20Environment%20Variables%20Step-By-Step%20Easy%20Guide%20-%20Like%20Geeks.htm#Global-variables)
* [**2 Local variables**](file:///C:\Users\CyberNode\Desktop\unix\Learn%20Linux%20Environment%20Variables%20Step-By-Step%20Easy%20Guide%20-%20Like%20Geeks.htm#Local-variables)
* [**3 Declare Local Variables**](file:///C:\Users\CyberNode\Desktop\unix\Learn%20Linux%20Environment%20Variables%20Step-By-Step%20Easy%20Guide%20-%20Like%20Geeks.htm#Declare-Local-Variables)
* [**4 Declare Global Variables (Export variables)**](file:///C:\Users\CyberNode\Desktop\unix\Learn%20Linux%20Environment%20Variables%20Step-By-Step%20Easy%20Guide%20-%20Like%20Geeks.htm#Declare-Global-Variables-Export-variables)
* [**5 Persisting Environment Variables**](file:///C:\Users\CyberNode\Desktop\unix\Learn%20Linux%20Environment%20Variables%20Step-By-Step%20Easy%20Guide%20-%20Like%20Geeks.htm#Persisting-Environment-Variables)
* [**6 Default Environment Variables**](file:///C:\Users\CyberNode\Desktop\unix\Learn%20Linux%20Environment%20Variables%20Step-By-Step%20Easy%20Guide%20-%20Like%20Geeks.htm#Default-Environment-Variables)
* [**7 System Environment Variables Paths**](file:///C:\Users\CyberNode\Desktop\unix\Learn%20Linux%20Environment%20Variables%20Step-By-Step%20Easy%20Guide%20-%20Like%20Geeks.htm#System-Environment-Variables-Paths)
* [**8 Variable Arrays**](file:///C:\Users\CyberNode\Desktop\unix\Learn%20Linux%20Environment%20Variables%20Step-By-Step%20Easy%20Guide%20-%20Like%20Geeks.htm#Variable-Arrays)
  + [**8.1 Remove elements**](file:///C:\Users\CyberNode\Desktop\unix\Learn%20Linux%20Environment%20Variables%20Step-By-Step%20Easy%20Guide%20-%20Like%20Geeks.htm#Remove-elements)

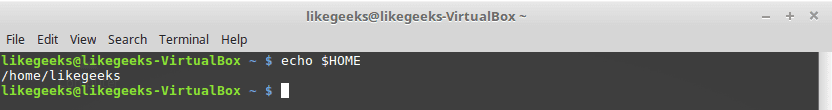
**Global variables**

The Linux system sets some global environment variables when you log into your system and they are always CAPITAL LETTERS to differentiate them from user-defined environment variables.

To see these global variables, type printenv  command:



As you can see, there are a lot of global environment variables, to print only one of them, type echo command followed by **$VariableName**. Ex: to print HOME variable type echo $HOME .



**Local variables**

Actually, the Linux system also defines some standard local environment variables for you by default.

To view the global and local variables for the shell you are running and available to that shell, type the set command.

**Declare Local Variables**

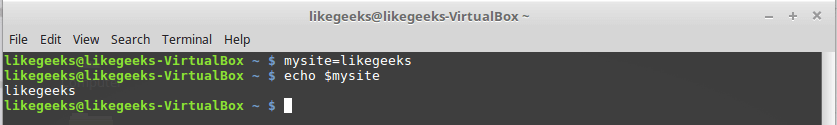
To declare your own environment variables directly from the shell, type variableName you want, followed by an equal sign and the variable value **WITHOUT any spaces:**

mysite=likegeeks

And to print the variable value, use the echo  command:

echo $mysite

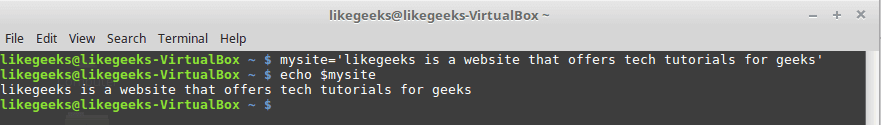
Sure enough, it prints likegeeks.



OK, what if your variable is more than one word; maybe a long string, you can put the string between single quotations:

mysite='likegeeks is a website that offers tech tutorials for geeks'

And if we type echo $mysite



If you forget the single quotation, the shell will assume that the second word is another command and will show an error.

As you can see, I use lower case characters for my variable not uppercase and this is recommended  NOT required, this helps you distinguishing your environment variables from the system environment variables.

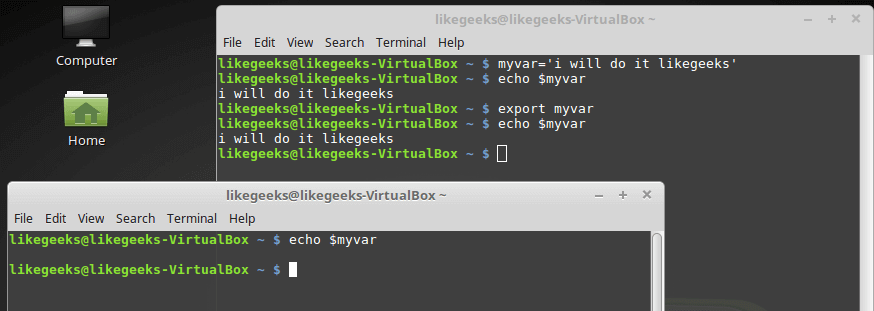
Once you have set your local variable, it will be visible in the currently running shell scope and that means if you start another shell window the variable will not be available in that new window.

**Declare Global Variables (Export variables)**

To declare a global environment variable, you have to declare a local environment variable then use the export command like this:



|  |  |
| --- | --- |
| 1  2  3  4  5 | myvar='I will do it likegeeks'    echo $myvar    export myvar |

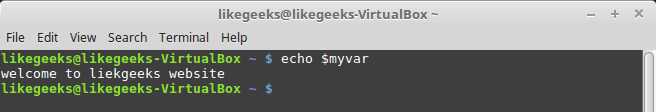


As you can see, I don’t use dollar sign with the export command.

But there is something, when I close the shell and open it again the variable is gone, so how to make it persistent?

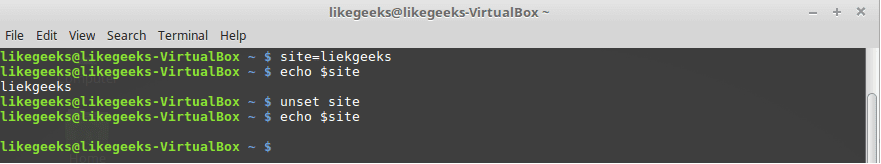
**Persisting Environment Variables**

Just edit $HOME/.bashrc and type  export myvar='welcome to likegeeks'  and save the file:



**Removing Environment Variables**

This can be done by using the unset command:



**Default Environment Variables**

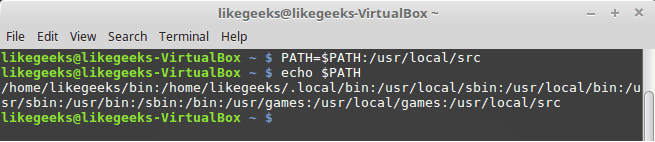
As we know, the system defines some variables for us, one of those variables is the PATH variable, this variable holds some paths.

These paths are the default paths that the shell uses to look for a command when you type it in the shell.

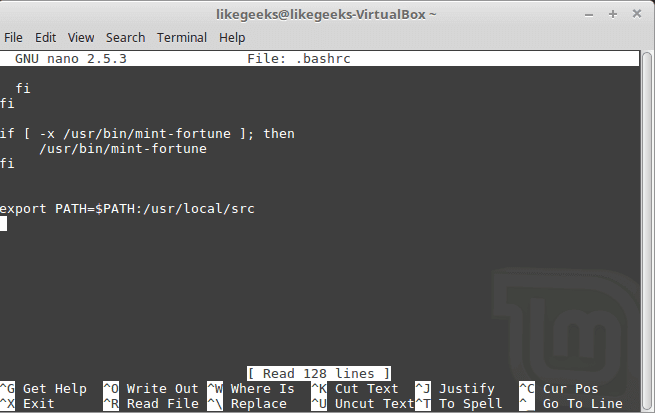
**Modifying the PATH Environment Variable**

If you add a folder path to the PATH variable, the shell will look in that folder for any executable to run when you type any command.

Just append the path variable followed by a colon and the new directory like this:



And if you want to persist the PATH variable, you have to edit the .bashrc file and add type it like this:



There is **a useful trick but RISKY** that some sysadmins do which is adding a period . to the path variable. By doing this, the shell will search for executables in the current directory you are in wherever you are.

This is relatively risky, you might give the attacker the opportunity to run a malicious script or malware in his current directory, so if you do this trick, you should know what you are doing.

**System Environment Variables Paths**

You can start a bash shell with one of the following ways:

* Login shell.
* The interactive shell.
* The non-interactive shell.

**login shell**

When you log onto the system, the search for those startup files to process the commands from them:

* /etc/profile
* $HOME/.bash\_login
* $HOME/.bash\_profile
* $HOME/.profile

/etc/profile runs on every startup with every user, the other 3 files run for every specific user, you can call them user specific environment variables.

**Interactive shell**

When you go to the rescue mode, this is the interactive shell.

If you start an Interactive shell, the system will not look for /etc/profile but instead will look for **.bashrc** in your **HOME** directory.

**Non-interactive shell**

This shell runs by the system itself to run a shell script.

Users can customize the above-mentioned files to include environment variables, just edit the desired file and type the variable you want and save it.

**Variable Arrays**

You can store many values as you want in one environment variable which is called array.

myvar=(first second third fourth)

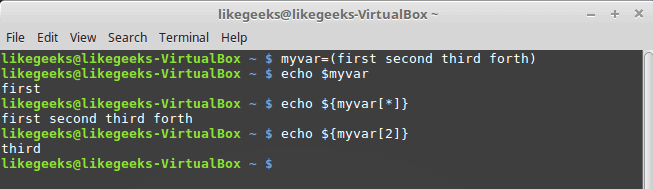
Now if you check the value of that array using echo command, you will find it returns the first element only.

To get a specific element, just reference it by its position, and the positions start from zero so to get the third one, type it like this:

echo ${myvar[2]}

To display the entire array type asterisk instead of a number:

echo ${mytest[\*]}



**Remove elements**

You can remove an element of the array using the unset  command:

unset mytest[2]

Or you can remove the whole array:

unset myvar