**Linux Environment Variables**

Before we go on reading about environment variables in Linux, I would like to make a not that around a month back “**I was a hardcore windows supporter**”, Though it is gospel truth that developers must use L**inux,**I never supported the fact (windows user “GUI Lover :P”). A few days back don’t know what came to mind, I shifted to Linux (also installed KDE “GUI Lover :P”), And trust me when I say this **Linux**is love. It gives you a lot of features to handle your machine at the core, Configure it according to your needs. One such feature which I found most useful is Environment Variables.

**The flow of the article will be like:**

* Understanding the jargon (Introduction)
* Local vs Global environment variable
* Linux system scripts
* Persisting environment variables
* Linux system environment variables

**Understanding the jargon (Introduction)**

Everyone knows what are variables, Data stored somewhere in memory when referenced by some indicator (variable name). Environment variables are none other than variables, It’s just that data stored in that variable is what describes environment behaviour. So now I think it is clear that why environment variables are so useful and interesting, One can control environments behaviour just by setting, altering, deleting several environment variables. Before we dive into playing with ENVs (environment variables) it is important to understand where to set them. All the play with ENVs will be done in a beautiful place known as the command line. One can access command line using Terminal.

**NOTE:** Linux environment has two kind of terminals one is Login(This kind of terminal is started when you login on startup) Another is Interactive(This is default terminal type and whenever you access terminal on your system it will be interactive). More on this is discussed later in the article.

**Local vs Global environment variables**

Coming from a coding background everyone knows about scope of a variable, Scope limits the usage of a variable in a particular section/block of the program, In simpler words a variable is **defined** only in a particular block outside that block it is **undefined,**Similarly Linux environment variables can take up any one of the Global or Local scopes. We already know that we need the command line to define ENVs, So what can be a local scope, or let’s first start with **global scope ENVs**:

As the name suggests, If an ENV is defined as a global scope variable then it can be used by any process, shell script also the command line itself, By “use” what I mean is its value is defined for all the programs or scripts run in that terminal.

**How to define a global ENV?**

1. Open any terminal.  
2. Type is export <varname>=<value>  
**Note**: There is no space between name, =, and value. If you give a space Linux will consider each of the three as a separate command, thus raising an error.  
So, now your variable is set. Next comes how to use it.

**How to use a global ENV?**

Using an ENV is an even simpler task, Just place a “$” before the variable name (no space between “$” and the var name), And Linux will take care of replacing it with its value.  
**Eg:** Type on the command line where you set the variable echo $<varname> .  
Also, you can use it in any script with the same mechanism.   
**Note**: echo command is just an example, environment variables can be used in the same way with any Linux command.

**NOTE:**Global ENVs are global only in the context of a terminal from which it is defined, Hence even being global ENV will not be valid for a terminal other than the genesis terminal of the ENV.   
For achieving this kind of use case we have to do something known as persisting the ENV, This is what we will explore in the last section of this article.

**Next comes Local ENVs,**

We have explored Global ENVs so what can be the constraints for a Local scoped ENV?   
You guessed it right, Local scoped ENVs can only be accessed by the terminal itself and not by any program or shell script even if the latter is started by the same terminal for which the local scoped ENV is defined.

**How to define a local ENV?**

1. Open terminal.  
2. Type in <varname>=<value>

Notice that defining the local scoped ENV is same as defining a global ENV without “export” clause. Other syntactic rules apply similarly.

**Note:** Accessing local scoped ENV is done the same way you access a global scoped ENV.

**Linux System Scripts**

Besides having environment variables, Linux also has some scripts or say some files which are used to control Linux environment behaviour. These scripts control Linux behaviour from boot time all the way to startup and run time. So what goes in these scripts, You can give normal Linux commands and also complex shell scripts in these files.   
In this section, we will go through some of these files which can prove to be very handy from day to day Linux use.

**NOTE:** These files are none other than normal shell scripts.

**Files that affect behaviour for every user:**

1. **/etc/bash.bashrc**: This file is read when ever an interactive shell is started (normal terminal) and all the commands specified in here are executed one by one.
2. **/etc/profile :**This file is read every time a user logs in, Thus all the commands executed in here will execute only once at the time of user logging in.

**Use Cases:**  
\* To specify global/local environment variables.  
\* Shell configuration, like colour profile etc.

**Files that affect behaviour for only a specific user:**

1. **~/.bashrc :**This file behaves the same way /etc/bash.bashrc file works just that it is executed only for a specific user. If you want to create an environment for yourself go ahead modify or create this file in your home directory.
2. **~/.profile :**This file is same as /etc/profile difference comes the way it is executed, This file is executed only when a user in whose home directory this file exists, logs in.

**Persisting Environment Variables**

We have already seen how to set global as well local ENVs, also we have read bout some Linux system scrips or files. Now the only part left for you to set and use ENVs creatively is how to make an ENV persist. Recall what we read in last section, We read about some scripts which are executed when ether we login or every time we open an interactive shell.

Bazinga! We figured out how to persist our ENVs, just put the commands which we have already discussed in one of these files and boom you have persisted your ENV, as these files will be executed when ever you access a shell.

Some conventions:

* Global environment variables are set in “.profile or /etc/profile” files as global variables can be accessed by any process or sub process activated by the login terminal.

**NOTE:** GUI which Linux provides is a process of the Linux login shell. Thus global ENVs can be accessed by any process we start on our GUI.  
Thus putting global ENV command in any of the “profile” files is a good method to avoid any redundant ENV set.

* Local ENVs can be set in both “bashrc” or “profile” files but it purely depends on the use case. If you want your ENVs to be accessible in your terminal (interactive) then it is to be defined in “bashrc” and not in “profile” files, as being a local ENV it will not be accessible by any other process started by login shell.

**NOTE:**Any customization command example “alias” can be set in “.bashrc or /etc/bash.bashrc” as most of us work on interactive terminals.

**Example of ~/.profile file:**

**Linux system environment variables**

In the start of the article we discussed that Linux uses ENVs to drive it’s behaviour, In this section, we will learn about some of these.

**NOTE:** Below mentioned ENVs are **global**.

* **$PATH**This ENV stores the path of all the directories which holds binary files you want to execute just by specifying the name of the file and not by relative or absolute path.  
  This ENV comes very handy when you need to switch between directories and still execute the binary file in some other directory.

**NOTE:**  
$PATH ENV can store various paths of the directories separated by a colon.  
eg: “**/bin:/usr/bin:/usr/sbin**”

* **$USER**This ENV stores your username. This can be used in shell scripts for various reasons.
* **$HOME**This ENV stores the absolute path to your home directory. This turns out to very handy when you need the path of your home directory in shell scripts.
* **$SHELL**This ENV stores the path to your default shell bin file, It maybe “bash”, “tcsh” etc.

Some other commands related to ENVs

* **printenv:** Lists out all **global** ENVs set.
* **set:** Lists out all the ENVs (**global** as well **local**)

**Conclusion:**

Linux is a developer friendly OS where you get to know how things actually happen. Environment variables prove to very helpful when used at the right place and with right value, Also persisting an ENV is another under the hood magic that happens, but now we know how to perform the magic. ;) :)  
Playing with Linux :)