2.5.1 Environment Variables

Click one of the buttons to take you to that part of the video.

Environment Variables 0:00-0:31

In this lesson, we're going to discuss the role and function of environment variables. Understand that whenever you start a shell session, several different variables are used to define critical parameters that the shell needs in order for it to run properly. Because these variables define the shell environment, they're called environment variables. Before you can understand what an environment variable really is, you first have to understand what a variable itself is.

Variables 0:32-1:44

The best way to understand the concept of a variable is to envision a bucket. In this bucket, you can store stuff, whatever it happens to be. Imagine this bucket has a label assigned to it. In this case, I've written on this bucket with a marker. The name of the bucket is MYVAR. The bucket itself is empty. I can put anything I want in that bucket. I could fill it with sand. I could, instead, fill it with rocks. I could, maybe, put dirt in it. I could even put water in it if I wanted to. If this bucket is already full--I've already filled it with something like water--I would first have to empty it out before I could fill it with something else, like dirt. That's essentially how a variable works on a Linux system. It's an area in your system's RAM that's reserved to store whatever values you decide to put in it.

Basically, it's like a bucket in memory. It's identified with a label. And just as you must empty out a real bucket before you can pour something else in, you have to empty out a variable before you can assign a new value to it. On a Linux system, you can define and use two different types of variables. The first are user-defined variables, and the second are environment variables.

User-Defined Variables 1:45-2:06

User-defined variables are just that. They're buckets in memory that you create for yourself. You assign user-assigned variables with the label of your choice, and then you fill them with whatever contents you want. This is commonly done when writing your own shell scripts. For example, you could write a script that prompts a user for a particular piece of information, and then you store that user's response in a user-defined variable.

Environment Variables 2:07-2:58

Environment variables, on the other hand, are initially created, named, and populated by the Linux operating system itself. As the name implies, environment variables are used to configure your system's computing environment. Environment variables are accessed and used by all of the applications and services that you run on the systems, even those that you run from the shell prompt.

Using the information stored in your environment variables makes the programs much more flexible and much more robust. For example, almost all distributions automatically define the SHELL environment variable. The SHELL environment variable stores the full path--the executable file--used to run the shell. And if you're using the Born-Again shell, Bash, by default on your Linux system, then the value of the SHELL variable is /bin/bash.

Commonly Used Environment Variables 2:59-3:09

With this in mind, let's review some of the environment variables that are commonly used on most Linux distributions. Not all of them will be used on all distributions, but most of these will be used on most distributions.

CPU and DISPLAY 3:10-3:33

The first one is CPU. This specifies the type of CPU installed in the system. We also have the DISPLAY environment variable. This is used to specify the location where your X Window graphical display should be set. And by manipulating the value of this variable, you can cause that your graphical environment not be sent to a local monitor, but be sent to a monitor on some other system somewhere else in the network.

HISTFILE, HISTSIZE, and HOME 3:34-3:56

We also have HISTFILE. This specifies the path to the command history file that the bash shell should use to save the list of commands you enter at the shell prompt. We also have HISTSIZE. This specifies the number of commands that you will save within your command history file. We also have HOME. HOME specifies the full path to the current user's home directory.

HOST, HOSTNAME, MAIL, and MANPATH 3:57-4:12

We have HOST and HOSTNAME. These variables contain the hostname of the system. We have MAIL; this contains the path to the user's mailbox file. Some distributions use this variable (MANPATH) to specify the path to your man documentation files.

OLODPWD 4:13-4:42

We also have the OLDPWD environment variable. If you have opened up a shell prompt and you have changed directories, then after you changed directories, that path that you were in before you changed directories is saved in OLDPWD. So, basically, OLDPWD contains the last directory that you were just in. The PATH environment variable contains a list of directories to be searched when you're running a command from a shell prompt.

PS1 4:43-5:05

The PS1 environment variable contains all of the characters that should be used to define what your shell prompt should look like. By manipulating this variable, you can specify whether or not you want your username displayed in the shell prompt, whether you want the hostname displayed in the shell prompt, whether you want the current time displayed in the shell prompt--whatever it is you want to do. And then we have PWD, which contains your current working directory.

The echo Command 5:06-6:30

There are several different commands you can use from the shell prompt to view the values currently assigned to your environment variables. For example, if you need to see the variable assigned to one single variable, you can use the echo command as shown right here.

The echo command simply writes text on the screen. And, by default, whatever you send to the echo command to display is treated as literal text. If, on the other hand, we specify that it echo a variable, instead of writing HISTFILE on the screen, the dollar sign ($) at the beginning of the variable tells echo that the text that follows is not literal text, but is actually the name of a variable. Instead of just writing HISTFILE on the screen, echo will go and grab whatever value is currently inside of that variable--the bucket HISTFILE. It will pull it out and write it on the screen instead. In this case, you can see that the value of the HISTFILE environment variable on this system is set to /home/ksanders/.bash_history. Using this environment variable, my shell knows where it should save all of the commands that I enter at the shell prompt.

The echo command works great for viewing environment variables, but it has two key drawbacks. The first one is it only displays one environment variable at a time, and there are a lot of environment variables. And the second issue is the fact that you have to know the name of the environment variable.

The env Command 6:31-6:54

If you want to view lots of environment variables all at once, or if you're not really sure what the name of the environment variable you want to work with is, you can instead use the env command, as you see right here. The env command displays all your environment variables along with their current values. You can see the syntax is the name of the variable, an equal sign (=), and then the value being assigned to that variable.

Risks of Changing Environment Variables 6:55-7:51

Before we go any further, I need to point out that most of the default values assigned to environment variables on a Linux system work great, so you need to be very careful before you change any of these. In fact, there are some environment variables that you should not change. For example, if you were to change HOST or HOSTNAME to a different value, it could cause problems with many of the services that are currently running on your system.

However, there are times when you do need to change the value assigned to a particular environment variable. For example, you may need to add an additional directory to the end of your PATH environment variable so you can run any of the executables in that directory without having to specify the full path to that executable. Or maybe you need to edit the DISPLAY variable in order to configure your X Window System to send its display to a remote computer somewhere else on the network. Likewise, you may want to alter the shell prompt to display different information.

Change Environment Variables 7:52-8:31

To do these tasks, you need to change the value assigned to an environment variable. It's actually pretty easy to do. At the shell prompt, all you have to do is enter the name of the variable, an equal sign, and then the value that you want to assign to that variable. It's important to note that if you do this, if you reset the value of an environment variable, then that new value you just assigned will be applied only within the current shell session.

For example, if I were to set the value of my PATH environment variable in one window and then were to open up a second terminal window and access a second shell session, I would find that the change that I made in the first window is not applied in the second window.

Persistent Environment Variables 8:32-9:10

In order to make that assignment apply to all shell sessions, you need to export the new value of the variable. To do this, you enter —˜export variable name' at the shell prompt. After you do this, the new value assigned to that value will remain available to all other shells, including any sub-shells that are created by the current shell.

One problem you will encounter in this process is the fact that any change you make to the environment variable at the shell prompt will be lost after the system reboots. If the change you just made needs to be persistent across system restarts, then you need to edit one of your bash configuration files and add the variable assignment command to the file.

Bash Configuration Files 9:11-10:00

Which file you need to add it to depends upon which distribution you are using, because some distributions will use one set of bash configuration files, other distributions will use other bash configuration files.

If you add the variable value assignment to the /etc/profile file, then that change will be applied to all the users on the system. If that's what you want to do, that's great. But there may be situations that you only want to make the change to the environment variable for a single user. If that's the case, then you need to use the appropriate bash configuration file found in that user's home directory. You'll have to take a look at the system you're using and figure out which of these it is you need to make the change to.

Essentially, that's how environment variables work. In this lesson, we talked about what environment variables are.

Summary 9:58-10:11

We talked about how to view the values assigned to your environment variables. We talked about how to set new values for your environment variables. We ended this lesson by talking about how to make environment variable changes persistent across both shell sessions and across system reboots.
