

System Commands
Online Degree Programme
B. Sc in Programming and Data Science
Diploma Level
Prof. Gandham Phanikumar
Department of Materials Engineering
Indian Institute of Technology- Madras

Lecture 5
Shell variables

Welcome to the session on shell variables. In this session will be learning about how to print the variables that are defined and made available in the shell that we are using in the terminal environment. Why do we need to know about the shell variables usually it is possible to communicate between two processes for tiny bits of information very efficiently using shell variables.

It does not require you to write or read file system which could be intensive in terms of the operations and often there are also security concerns that some information that you write to the file system may be visible to such other processes which are able to read the file system. Shell variables are available only within the shell or its child processes if you enabled the export option and thereby the information that is stored in shell variables is fairly private to the shell.

Shell variables are also a means by which you could store information as an intermediate form for you to process further. Certain processes such as commercial software that expect the license server or the port number on which the license is available and so on may be looking for certain shell variables to have that information. And therefore even when you do not use the thermal environment very extensively it could be necessary for you to know about the shell variables.




So, to be able to what are the variables defined in the shelf what are the values that they contain how to define those variables and modify them would be part of our exercise and today we will see how to just display what are the variables that are available for us in the shelf.

(Refer Slide Time: 02:06)

echo

Print strings to screen
`echo hello, world`

Print values of variables
`echo $HOME`



In this session we will be extensively used the command echo. The echo command prints out the value of a shell variable or a string that is supplied on the command on screen for example as you can see you could print hello world using a single line like echo hollow world and you will see that the string hello world will come out of the screen. You could also print the values of certain variables that are there in the shell environment by using the echo command.




And we would notice that there is a convention to store the names of variables in the shell every shell variable will start with a dollar.

(Refer Slide Time: 02:47)

Frequently used shell variables

- \$USERNAME
- \$HOME
- \$HOSTNAME
- \$PWD
- \$PATH

`printenv, env, set`



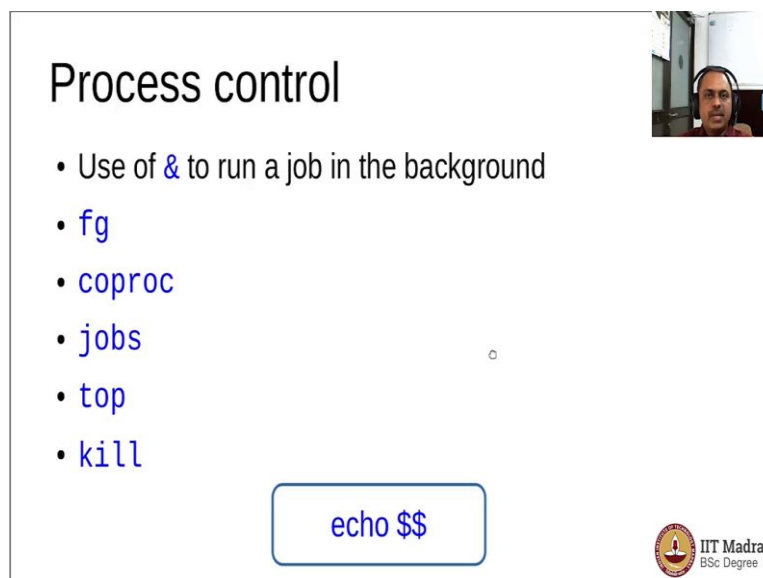
There are certain frequently used shell variables such as the user name the home directory the host name are the name of the machine on which this particular shell is

running and the present working directory which will keep changing its value as you keep navigating through the file system. So, the path variable contains a list of directories which will be searched for when you type a command.

So, there are certain commands available such as `printenv` or `env` or `set` which can help you see what all the shell variables that are already available and defined in the shell that you are using. There are some special shell variables such as the dollar zero which will give you the name of the shell and then double dollar which will give you the process id of the shell we will see what does it mean by a process id shortly and dollar question mark which will give you the exit code of the program which just executed.

Dollar hyphen will tell you what are the flags that were set by launching the batch shell that you are using currently.

(Refer Slide Time: 03:50)



The slide is titled "Process control" and lists several commands for managing background processes:

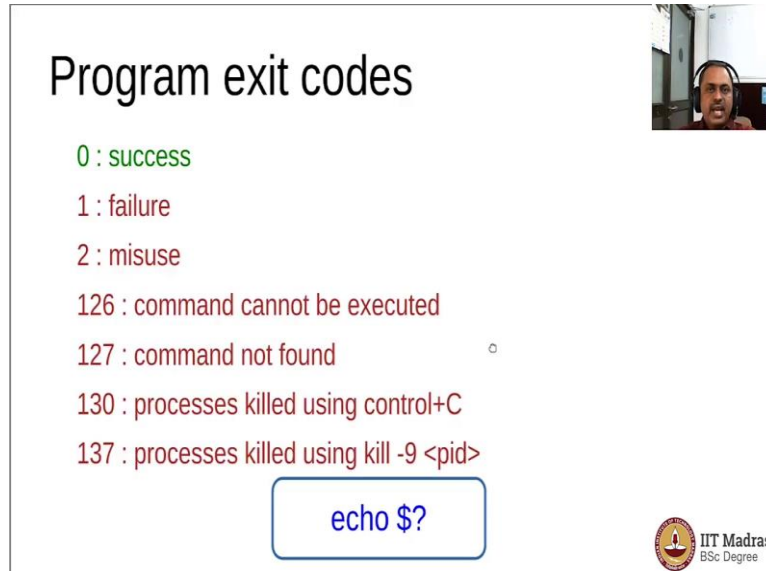
- Use of `&` to run a job in the background
- `fg`
- `coproc`
- `jobs`
- `top`
- `kill`

Below the list, there is a terminal snippet showing `echo $$` inside a rounded rectangle. In the bottom right corner, there is the IIT Madras BSc Degree logo.

We will also see certain process control whereby you could run a program in the background and then you can bring it again back to the foreground you could run a command while also being able to go on to use the shell. You could also list what are the programs that are running in the background using the `jobs` command you could see what are the programs that are hogging the memory or CPU utilization in your computer and it is refreshed every second you could look at it using the `top` command.

You could kill some processes which are owned by you using the kill command. So, we would actually start exploring the job control starting with the special shell variable double dollar.

(Refer Slide Time: 04:33)



Program exit codes

- 0 : success
- 1 : failure
- 2 : misuse
- 126 : command cannot be executed
- 127 : command not found
- 130 : processes killed using control+C
- 137 : processes killed using kill -9 <pid>

echo \$?

IIT Madras
BSc Degree

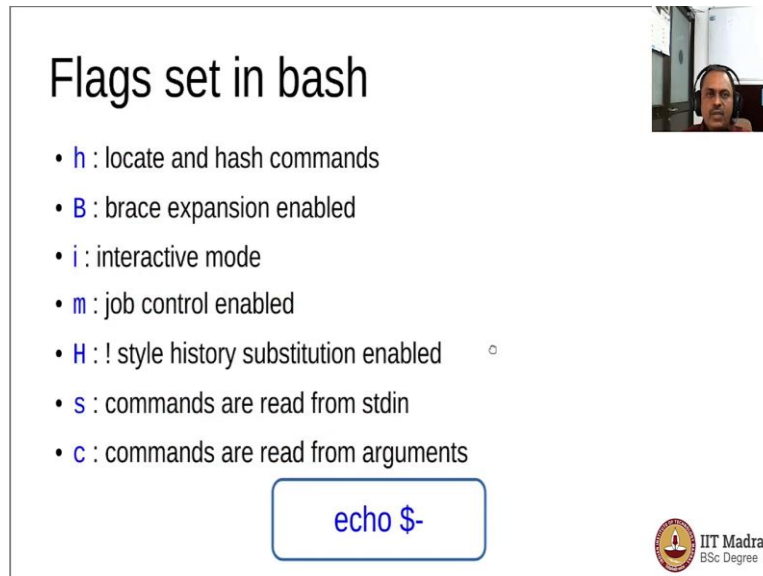
When it comes to the program exit codes it is a matter of convention that the exit code should convey in what way the program has ended. And exit code like zero would convey that the program has been successfully executed and that is universal it would evaluate to true if you were to use it as a bullying variable while checking whether the command has successfully executed or not.

And values one or above are used to convey certain amount of value in running that program. The value of one as written code would mean that the command has failed and whatever is the reason would be displayed on the screen and exit code like two would also mean failure but for slightly different reasons sometimes for misuse of shell bulletin or perhaps when the permissions are not adequate.

Written code of 126 is used when the command could not be executed perhaps because it is not an executable and if you try to run a command and it could not be located within the list of directories specified by the part variable then you would have an error code which is 127. The exit code 130 is used when the program that was running in the foreground has been killed using the control C operation.

If a process exited because it was killed by using the kill command using the option -9 then it would have an exit code of 137. If the exit code happens to be more than 256 then the modulo of value with what 256 would be reported as the exit code. So, that the exit code is always having a value between zero and 255. We will explore each of these exit codes by running appropriate comments to ensure such an exit code would be reported and thereby you will understand how the written codes are coming up.

(Refer Slide Time: 06:25)



Flags set in bash

- **h** : locate and hash commands
- **B** : brace expansion enabled
- **i** : interactive mode
- **m** : job control enabled
- **H** : ! style history substitution enabled
- **s** : commands are read from stdin
- **c** : commands are read from arguments

echo \$-

IIT Madras
BSc Degree

When a shell is launched it would be launched with certain options that are already set and some of these options are like the option h which is to locate and hash or memorize the comments that are being executed. The option capital B to enable what is called the brace expansion we will see a demonstration of that. This small letter I to indicate that it is in an interactive mode.

The small letter m to indicate that the job control has been enabled that is the jobs can be taken to foreground or background etc. The option capital H is to enable the history operations such that using the bank or asterisks and then a numerical value you could actually repeat the comment that was executed slightly in the past. The option small s would be used when the comments are supposed to be read from the standard input by that what we mean is when we type using the keyboard.

The option small c is used when the comments to be executed are to be read from the argument passed on to the batch beyond what is given as options. We will use these options to understand the output of dollar hyphen special shell variable.

(Video Start: 07:42)

Let us get started with the shell variables as we have seen echo is the command. So, let us see what type of a command that is. So, I will type man echo to know more about that and you see that the man page says that echo command is to display a line of text. And so let us try that out. So, we would say hello world and you would see that the string has been reproduced.

Now I would type hollow world with multiple spaces in between and you will see that you will still get only one space which means that the echo command is looking at what is supplied as an argument and picking up words from there and using a single space as a delimiter and then displaying them on the screen. Now if you want to actually have more spaces then you would have to enclose the string in quotes.

So, you can now see that the number of spaces that we gave has been kept intact. So, it actually is a lesson that the echo command when it is applied with an argument that does not contain the quotes then it would actually treat the words as separate values which would be taken as a list and then they will be displayed with one space between and therefore if you provide multiple spaces then they will be ignored.

What happens when you do not match the quotes properly. So, you would see that you are being prompted with a different prompt string which is just the right angle bracket it means that the shell is now expecting you to finish the command with the matching double quote and until that point it would actually keep asking you. So, I would actually ask now I can actually give some more input and if you keep asking that.

So, I closed a single quote but that does not help because we have started a double quote and we have to finish the double quote. And now you see that the command has been completed but meanwhile you have noticed that the string the entire string has been read and you have all the four lines coming up onto the screen. So, this is also a lesson to learn two things one is that you can in fact enter a multiline input to the echo command.

In case the shell was expecting you to complete the command it would prompt you on every new line using the right angle bracket until you complete the command as per the quotes that are matching. So, let us try with the single quotes and they would also work

and here now the matching quotations required are single quotes. So, I would actually finish the double quote but that does not finish the command.

So, now you see that the single quote when it is given the command is completed and therefore you now have all the three lines being displayed. So, which means that whatever is the first quote that you started that has to be finished and along the way we have opened another pair of quotes that does not matter. So, let us clear the screen now. So, the echo command that way is very nice in reproducing whatever text that you are giving either it is a single line or multiline.

And usually in a shell script you would actually use it to display lines of text that would help in documentation. For example you may want to have a command which would actually display a line sometimes you may want to give some comments and sometimes you just need to display a plain text. So, you could use the echo command like the print of command in C language liberally. Let us clear the screen now.

Now you can use the echo command to also display values of shell variables and as we mentioned earlier every shell variable starts with the dollar symbol. So, the very first variable would be to display the user name the user who is logged on and you would see that the user name is displayed on the screen. You could also enclose the shell variable within double quotes and it would be interpreted and then displayed.

What happens if you use it in the single quotes? You see that the string that we gave was displayed as it is but it was not interpreted to give you the value of that particular shell variable which means now you see the difference between the single quotes and double quotes within the double quotes any string that comes with a dollar will be interpreted as a shell variable and if a value is available for that that value will be substituted and then printed.

And if you have the same thing in a single quote then it was it is not going to be interpreted. So, you could also mix certain other strings along with these shell variables within the double quotes. So, now you can see that we are trying to print a string followed by the value of the shell variable user name and it comes out quite. So, there are many more such useful shell variables let us look at each.

So, dollar user is also the same as the user name dollar home is the home directory let us look at the value of the shell variable pwd. So, it would have the value of the present working directory. So, if you type the command pwd in small letters it would have the same value and you could now change the directory and test it out. So, I go to let us say user and I will check ok.

So, you see that the value of the shell variable pwd and the output of the command pwd in small letters is the same and it would be the present working directory and you can navigate within the file system hierarchy and you can try it out at any location and it would be the current location that currently the command execution is being performed. You could always inquire the name of the machine using the host name shell variable and that would be the name of the machine as given in a file called etc host name.

And it would be a static file if the configuration was like that and it can also be made dynamic if you had the network settings so configured. Now let us say we want to print a string where we would some of the variables to be interpreted but not certain others which means that you do not have a choice to pick the single quotes or double quotes you need the double quotes because you want some of the variables to be interpreted but you want some of the variables to be escaped.

So, let us see what we mean by that. So, in this particular command you see that the host name and the user variables are both interpreted and I would like to interpret only one of them for whatever reason. So, we will realize that very shortly. So, let us see how that is possible. Now you see the difference in the second command we have used a backslash which meant that the dollar symbol is then not interpreted as the first character of a variable and therefore the entire word is then not interpreted as a shell variable and is displayed as it is.

Whereas the other variable did not have the escape sequence of backslash and therefore it will be interpreted and the value of that particular shell variable is displayed. So, you can now see that you can have a possibility to escape the interpretation. Now why is it useful? This is useful when you want to pass on the string to a child shell but you want the interpretation not to be done by the shell that is launching that particular command.

Now you may be curious to know how many shell variables have been defined in the shell that you are running. So, let us go print env would actually list a whole bunch of shell variables and I am scrolling with my mouse and you can see that each of these lines have an equal to sign separating the name of the variable and the value. And you could see that there are a lot of them and we will learn about these as we kept ourselves familiarized with the operating system as well as the command line environment.

Now you can get the same listing also using the command env okay there is the same listing that you are having and you could also get the list of variables set using the command set okay and using this command set. You would also see certain functions that have been defined to interpret what you are typing on the command line apart from of course the shell variables that have been defined.

The output is quite long so you may have to scroll quite a bit before you come to see the list of variables that were defined. So, now you see that there are a bunch of variables that have been defined and they are being listed by the output of set command. Let us look at the help man page on print env command and it says that you could print the all are part of the environment.

What do we mean by the part of the environment you could do like this print env and let us say you give the name of the variable without the dollar symbol and it would give you the value of that particular variable. You could also do that with anything else that has been defined. Now let us look at how we can escape the aliasing just the way we have escaped the dollar interpretation you could also escape an alias.

So, we will try that using the date command. So, we have tried that date command in the past and we have also seen that date command can take an option of capital R to print the date in the format usually used in the email communication. So, let us say I want to alias the second way of displaying the data for the default behaviour. So, I will do that by typing an alias date is equal to and date minus R.

So, I can then check what is the alias for the date I need to tell in what way the date has been aliased and now if you type the date command the output would be as if the option -

R has been given. So, you could now see the difference here the date command output here and the date command output here. So, the difference is coming because the `l a s` is now active. So, this date command you could actually ask where is it and it is a user bin date.

So, you could actually execute it directly using the absolute path. So, I could type this and you would see that the output is without the alias namely the default format that was defined for the date command. And you could also navigate to the folder where this particular command is located and you can execute it from there also. So, I would go there go to slash user bin and I am giving the relative path of the date command dot is for the current territories for dot slash date is the name of the file that is going to be executed.

And you would see that it executes the same way as if you type the date command without anything before it. Now you can now let us say I want to type the date command but I would like to escape the alias that I have kept. So, for that you can actually use the same backslash the way we have escaped the dollar symbol in the previous example.

So, if you do that then the aliases are all escaped and you would run the command with its default behaviour. So, what we have learned is that there are multiple ways of executing commands. You can execute a command by going to the location where it is placed and using the relative path for the particular file. You could execute it directly by typing the name of that command wherever you are in the file system you can be in the user bin or for example you can be in the home directory just typing it out.

Or you could also type it with a backslash in front to indicate that you do not want to use the alias. Now when you type a command how does the shell find out where is it? This is actually obtained by looking at the path variable. So, there is a variable called path. So, this path variable contains a list of directories here you can see that the very first directories user local sbin okay after that it is user local bin and then it is user sbin and then user bin and then slash sbin and then slash bin and so on.

Which means that when I type a command shell would look up each of these directories in the sequence from left to right to identify where is it which directory contains a

particular executable and then executed from there. Now let us look at the special shell variables that we have mentioned the first the dollar zero. So, if you see the output of dollar zero is the shell name that is currently running the shell command.

Now we can test it out by launching such an other shell which may be present in the particular operating system and trying out the same thing. So, I would like to now launch and let us say hey k shell. So, if it was available it would open and it has opened up and I would echo dollars zero and it says that it is a case that is running I come out of it using exit command. The next special shell variable that we were discussing is the double dollar and the double dollar basically gives a number on the screen which is the process Id.

What it means is that there is a number given for the process called the batch that is currently running and interpreting the commands that we are giving. So, how to know what are the processes that are running? It is a ps command that will help you to know that. So, let us look at the man page of the ps command. So, ps command says that it would report a snapshot of the current earning processes.

It says snapshot because the processes keep changing new processes get launched certain old processes would finish and close. So, ps command can tell you what are the processes that are running right now? And if you see here it is running two commands batch as well as the ps command which is what we have just now run. And you could actually see the output of the ps command very nicely to know which process has launched which child process by using the option double hyphen forest.

So, it would actually display these two characters to indicate which shell or which process has launched which other process. And the integers that are placed in the first column under the header PID are the process Id numbers. So, when you type echo double dollar the number that you get is the same thing as what is listed again is the batch process when you run the ps command.

Now that brings us to the discussion about the ps command you can actually see what all the processes running in the operating system right now by using ps and options like -ef and you would see that a whole bunch of processes are running. And if you scroll up and

you will see that the very first pid is assigned to a command which is basically the init. So, that is a very first process that was running.

And if you see ppid header that is a parent process Id. So, which means that which process I restarted this particular process is what is listed. So, if you look at the process id three it was launched by the parent with a process id two and that means that rcu underscore gp process was launched by a process called k thread b. So, like that you can actually identify which process has launched which other process.

So, the parent ideas are available for you to help with that. Now the ps command with the -f would actually give you the parent Id and you could also see that the ps -f command has the process Id X8615 and it is launched by a parent 3423. And 3423 is a process Id of a command batch which is a shell. So, you can actually see how the child process and the parent process can be identified using the output of the ps command.

So, some of the important options for ps are ps -f to see the parent id also ps -e to see all the processes that are running and you could combine them both ps -ef to list the parent id as well as all the processes that are running currently in the operating system. And you could also have the option --forest to see what all the processes that are running in a way that is easy for a novice user to identify.

(Video End: 26:32)