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

Linux process management

(Video Start: 00:14)

To learn the concepts of job control we would need a process that would run for at least several seconds. So, that we could try to stop it put it in the background or bring it to the foreground or even kill it. And as you start writing programs later on you may have processes that will be running for a long time but right now as a beginner you may need a very easy process that you could kill without the damage and one such process is called sleep.

So, look at the main page of the sleep command it is basically just a delay for a specific amount of time which is given in seconds. So, if I type sleep and then say three it would sleep three seconds and come back to control. So, you just wait for three seconds the control will be brought back and the prompt is available for us to run. So, if you did that for example for ten seconds then you will see that for about ten seconds you cannot do much because the prompt is not available for you to type anything and you have to wait for the control to be given back to you.

So, the control has come back. So, what this implies is that if you have a command that is running in the foreground for a long time but you need to have the control back to the prompt. So, that you can write something else on the commands line then many options you may want to kill the process if it was not important or you may want to suspend the process. So, that you want to do such another work and then again bring it back to action by bringing into the four ground.

So, you could try out of this and you may even have an option of running a command without actually losing control of the prompt by keeping it in the background but executing. So, let us try those out now I would use the commands called coprco sleep ten now what happens is that I have brought the command command prompt is now back to me and we are able to write some commandss and as in when the process is completed you would know because information is available on the screen to say that that particular command has been completed.

So, it says that the process has been finished So, coproc is a command to help you run a process without actually losing control of the command prompt to do other activities. So, when you run a command coproc and let us sleep and we'll give it for 30 seconds what happens is that when you type the ps command. So, you would notice that the sleep command as a ps command are child processes of the bash shell and you can now do something to the sleep command.

So, let us see what is this coporc command and you would see that there is no manual page available. So, we need to find out what type of command that is. So, I type type coproc I ask and it says that you say shell keyboard which means that it is a feature provided by the shell and we have already learned earlier that to learn more about any shell feature we should use the help command.

And the help commands definitely gives us some information it says that create a core process with the name that is being given as an argument and it should be run asynchronously but you are having the control of the prompt immediately. So, let us launch a command and then try to kill. So, I launched the sleep command which is running for thirty seconds and the process Id is displayed on the screen but I would also like to inquire it.

So, 9486 is a process Id. So, I can kill it by using a kill -9486 and you will see that it would actually lead to killing that process called sleep thirty. So, if you have a process that is running and you do not want it to run if you know the process Id and if you are the owner of that particular process then you can go ahead and kill it if it was safe. Now if you want to run the same process; while keeping it in the background rather than running it simultaneously you could do that with an ampersand sign at the end of the command.

So, if you type like sleep thirty and then an ampersand it means that you are going to put this commands in the background. So, now you see that the command prompt is available and it also tells you that this particular command is in the background So, now now you want to bring it back to the foreground and you can type fg and it would actually bring the command which was in the background to the foreground.

And now it is in the foreground and you do not have the prompt with you, you will have to wait for the command to finish and of course I do not want to wait. So, I could press control C and kill it. So, when the process is killed the command prompt is now again available to us. So, there are two ways of killing the process one is if it was running in the foreground you could type control c and if it was running in the background or in any other shell as long as you are the owner you could kill it using the kill command.

So, there are two ways of killing the process by knowing the process Id. So, therefore the knowing the process Id is quite important in managing the processes that are running on your computer. Nnow you may be curious to know if in the shell you have launched a certain processes and you have not paid attention to them and are they running in the background still.

So, there is a command called jobs that could actually tell you if there was any commands that is running in the background. So, let me run a processing in the background and type a commands called jobs and it tells and it tells that there is a process called sleep thirty that is running in the background. I run another command sleep forty five and it is giving me warming still you could actually go ahead and do that and you will see that there are two processes running the background.

And a ps minus minus forest would show you graphically how those processes are listed under the bash because bash is the parent processes the two sleep commandss are the child processes and of course the ps command itself is a child process of the bash. And as soon as the jobs are done you would have the display on the screen when you press the enter on the command line as a process of running various commandss. So, whenever the command is executed and completed you would see that the information about that completion will be displayed on the screen.

So, both the process are now finished. So, let me then type the jobs combined to see if there are any jobs in the background and you see that there is nothing in the background. So, what is what is this jobs command and let us look at it? So, that we press control L to clear the screen and you see that there is no manual page for that s, our suspicion would be perhaps it is a shell feature and as expected is a shell bulletin.

So, we had to then run the command help to know more about that. I am sure enough there is some help about the jobs command and you can see that it is about listing the active jobs that are running in the current shell. If you are curious about what are all the jobs or processes that are running in the computer there is a very nice commands top which can tell you what are the processes sorted in the decreasing sequence of the CPU utilization.

So, if you type top and you will see that the screen is going to be refreshed every couple of seconds and it would show you the process that is occupying the maximum CPU utilization on the top. So, if your computer is not responding well then in a shell you can actually open this to identify which command is running and occupying the resources. You would see that I am recording this particular video using the OBS studio.

And OBS is what is actually occupying 63%t of the CPU and it is at the top. Now you can come out of this by pressing q. So, I clear the string with control L. I can launch the top command and while it is running I can press q to come out of it graciously. However I could also come out of it by pressing control C. Control C is like interrupting and killing the process and coming out and we have done that just now.

So, when I am running this command top you could also do one more thing you could press control z. So, what happens when you press control z or control z is that the pros the process is suspended. So, if you press ps then you would see that there is a process that is still there it is not completed but it is suspended. Type jobs to see and it says that there is a job that is stopped and it is called top and we can do something with it.

Type fg and it would then come back to the front and it is again refreshing and working. Now if you want to come out of it graciously you would press q and come out of it. Now what this shows you is that you can actually suspend a program do something else and then again bring the suspended program back to the front and then come out of it elegantly whenever it finishes. So, this allows you to be able to switch between tasks while you are in the command line environment without actually coming out of the shell.

So, the command that we used to bring the background job to the foreground was fg. So, type fg and there is no manual page available and what type of a command is that is a shell bulletin we can ask for help of that and it says that move the job to the foreground. So, if it was a job that was running in the background it would be brought to the foreground. So, it would be an active on the screen.

So, let us just try this out by the command sleep itself. So, we will type sleep sixty. So, for one minute and let us say I put an ampersand here. So, it goes to the background okay and then I could do some commandss in between and now I want to bring back that commands to the foreground I can just type fg and you see that the sleep sixty is now in the foreground. And I can now not type commandss because the prompt is not available.

And let us say I am impatient I do not want to continue further I can press control z then it would be stopped and you can see that the prompt is back to me and I have not killed the process I have only stopped it just like suspension and let us say I now give time for it to finish up the work of sleeping for sixty seconds. So, I can then type fg and the job is again back to the foreground.

And let us say I do not want to make it finish all the time let us say I want to interrupt this program by using controls the account interrupted and thereby killing that. Let us look at the meaning of the shell variable dollar hyphen which is a very special variable because it tells you a lot more about the capabilities of shell that is currently running. So, depending upon the options that you give to the shell; while launching it its capabilities can be limited or expanded.

So, you see that there is something that is printed on the screen it says h i m capital B capital H and then small s the meaning of these is available in the main page for bash you see that the options here whatever is after the hyphen symbol are the options that we can see. So, the sea would mean commands are to be read from the argument supplied on the command line after options the I would mean that it is an interactive shell and so on.

Let us say we want to see a bash shell with less number of options configured then what is displayed here. Particularly let us say we want to launch a bash shell which is not an interactive mode. So, we could try that out now here like this. So, if I type a bash and I would type -c because I want to pass on a command and I wanted to return back to the parent shell. So, I would then do this equal you now see that the bash shell that was launched has less number of options and it is not interactive.

So, which means that when we ran this command child shell was created which had only one action to do namely eco or type out the value of the variable dollar hyphen and then it exit. The reason why we use the backlash is because while passing on this command to the child shell we do not want to interpret it already. So, we may now enhance this command to see whether it is a child shell that is responding to us by printing out those options hBc.

So, I combine multiple commands using the semicolon and you will see that what is responding to us is actually this bash because the output of the p s command is coming from there and you can see that it says child shall spanned by the shell that we are using right now how do I know. So, double daughter says what is the shell I am using right now. So, I am here and this shell has spawned the other shell which has actually then launched the ps command.

So, we may now want to inquire process Id of the child shell echo double dollar then echo. So, now you can see that the first command has given me the output which is the process Id of the child shell which is actually listed here 11213 and the second command is giving me the options with which the child shell was launched which is hBc it is not interactive and the third command is telling me the process listing as it is displayed here in the next three lines.

So, we have combined essentially three commands using the semi column and that is actually passed on to the child shell and we are able to see that you could control the options with which the bash can be launched. Let us use the history command to understand one of the flags with which the bash shell has been launched namely h for history and get the man page and it shows you that there is a possibility to run a particular command in the history by referring to its number after the bank symbol.

So, when you type history you would see all the commands that have been run in this particular shell in the last several days and if I use the bank symbol followed by a number that particular command will be run. So, here we have the date command with 1992 as the number. So, I type 1992. So, bank 1992 would mean the command date that is as per the history should be repeated and that is what would be repeated.

Of course it does not make sense if you have commands that are too small that you need to refer and type if you want to repeat a commands that is a bit long to type it is very useful. So, you can now see bank 2004 would repeat this particular command and you would see that that particular command has been repeated. So, it is pretty useful to have this feature in the bash which is available.

So, when you inspect the value of the special shell variable daughter hyphen then the capital H that is there indicates that a particular feature of referring to the history and being able to run a command from the history by using the numerical value of that particular history line is made available. A couple of words about the concept of expansion on the shell we will delve into that deeper a little later.

But for now let us look at the brace expansion because that was one of the options that came up with the list of flags when we looked at the output of echo dollar hyphen command. The capital B here corresponds to the brace expansion option is made available. Let us look at what does it mean. So, if you type with two double dots characters which are listed in the ASCII sequence will then be expanded as per our need.

And if you type like this what it it means is that you would get to the values from a to z typed out and you could actually configure this let us say from a to only l and also from capital A up to capital K and so on. You could also use them in combination with each other A to D and then a letter after that A D. So, you would see that all the combinations of A to D as a two letter combination would be listed.

So, this is what is called as a brace expansion. There are many other expansion that are possible the most common one is the wild card star. If you type echo star it would give you the list of all the files and directories in the current directory which would mean that the star expands to all the files in the current directory. If you would type capital D star it would list the directories that start with the D which will actually mean that the star would actually expand to what is possible given the first character is capital D.

So, if you type capital D small o and the star it would list only two of those folders because it would expand to all those files which match the pattern namely capital D small o followed by anything that would actually be possible in the list of files or directories in the current directory. Now if you want to launch multiple commandss on a single line it is entirely possible you could actually use the semicolon to separate of the commends.

So, let us say I want to type the ls command. There is some output. I have the date commands there is some output and then I have a command called wc -l slash etc slash profile which will tell me the number of lines in the file called slash etc slash profile. There are 27 lines now I want to run all these things in a single line. So, we could do that like this ls and then semi colon and the date and the semicolon and then wc -l slash etc slash profile and then the semicolon.

So, the semicolon acts as a separator between individual commands and you would see that in one go you are running all the three commands one after another in the sequence that is left to right. Now you could actually have spaces around the semicolon or not it is up to us. So, you could have like this for readability or you could also have it in a crisp manner or you could also skip the last semi colon and the output would be identical.

So, you can see that you can combine multiple commands on a single line using the semicolon which also means that as soon as the semicolon is appearing on the command line it means a new command has started. So, let us see what happens if you combine with it in echo command. Now you see that the first command has executed after that the semicolon is interpreted as a separator with another command.

So, therefore world becomes the command that is being asked for and there is no such command and therefore we have got an error. So, which means that when you type like this it is interpreted semi colon separating out this particular line into two parts the first part is one command which is echo hello and then the second part is just the command called world. So, that really is there will be an error because there is no such command available.

So, if you happen to type something like that then you will see that the first part executes as echo hello the second part executes as ls because ls is a command by that name and therefore that output is given and look at the difference between this and the line that I am giving you. So, without the semi colon the entire string is interpreted as a what is to be printed on the screen and therefore the output actually is very different.

So, this semiconductor is a very important special character interpreted by the shell as a separator between commands. Let us explore the exit codes or written codes in the bash environment. So, if you type dollar question mark then it would give you an integer on the screen an integer between 0 and 255 which indicates a written code of the previously executed command.

If it was zero it would mean that the previous command was successful and if there was an error in the previous command then it would have a value from 1 to 255 depending upon what type of an error that is. So, let us intentionally make commands that will not succeed and thereby inspect what would be the written code or error code. When a type ls slash blah there is no folder called blah in the root directory and therefore there should be an error and you see that there is an error.

So, we can go back to the root directory and look at the files or folders that are there in the root directory there is no folder or file called blah and therefore it should give an error. So, when I ls slash blah and then it has failed. So, now if I type echo dollar question mark then it would give me another court corresponding to the way the previous command has exited.

And therefore the eldest slash blah has exited with code two. So, the two indicates that it was having a failure. Now let us go to the root folder and try to create a file here which we do not have permissions for. So, I will try to create a file called file one here I do not have permission. So, therefore I get an error and now I could actually inspect what was the error code and you see the error code is one.

So, it means that zero is a code for success one is when the parameter is denied and two when there is a certain other error of a failure because such a path did not exist. Now I am in my home directory I create an empty file call file one and inspect the permissions and you see that there are no executable permissions for the empty file called file one and as we have seen with the date command you could execute it from the location where the file is present using the shortcut for the current directory as dot.

So, you could do like this and it would try to execute but there is no executable permission you would have an error. Now you could actually inspect what kind of an error code that was and it would be an error code 126. The error code as per the documentation is when the command has failed because the file could not be executed. Now let us say we are tripping a command and we made a typo and therefore such a file does not exist and therefore there should be an error.

So, let us say I am supposed to type date but I would create a type and type it as daet and it would take a couple of seconds to actually look through the system and then say that there is no such file and through a message saying that such a commands was not found. So, what was the written code for the attempt that we did just now? You see the written code is 127. So, the 127 written code is reserved for situations well the file was not found.

Let us say we are executing a program and we exited from it by purchasing control c then you could also see that the error code would be different one. So, what we would do we will try to run the command sleep thirty but we will not wait for thirty seconds we would try to kill it with control c and then try to see what was the written code and you would see the written code is 130.

So, again 130 is a written code reserved for situations where you have interrupted the process by pressing control C. Let me try a situation where the process is running in another location on the same computer and I am trying to kill that particular process and then we inspect what would be the status of the error court. So, the terminal app is pretty useful in that sense I have this icon here in the corner to launch a tab. So, we have got a new tab here. So, in the new tab I am running a command top.

So, you see that the top command is running on the new tab and in the old tab I can inspect what are the processes that are running in the computers. So, I type ps and you see that it does not list the top process because it is a separate shell however it is on the same computer. So, we can actually see it by using the -e command which is listing all the processes in the computer that are running and sure enough you can see that the top command was running here it has a process Id 12748.

So, I would go ahead and kill that process by passing on an option called -9 which is to basically kill that process okay when I do that and type ps -a I see that the process is no longer there and when I go to the other tab you see that the shell is now back to the control on the screen the top command is not running. So, which means that I could not inspect the status of the previous command and it says the it says the code is 137 which means that the exit code.

137 is reserved for processes that were killed not by pressing control c in the same shell but by sending the -9 option to kill from another shell. Let us explore situation where the exit code of a program is an arbitrary number but that we would use bash as a program. So the process Id of the bash that I am using is 15083. Now I would launch a bash shell now by passing on some commands.

So, one would be to print the process I d of the child process and then to list the processes that are running in the child process and finally to exit with a particular record which is three hundred as an illustration now you could see that from the output the parenthood is with the PID number 15083. The shell that we launched now is with the paid 15315 as you can also see that 15315 is a child shell launched by the parent shell bash with the id 15083.

We exited with the code 300 and if we inspect what was the exit code you will get the number 44 and this is because you are taking the model of width to get the exit code. We can try the same thing with any other number. So, let us run the same command and we give a big number and see what happens and you get an exit code that is 215. Now we want to confirm whether this is ok.

So, we would use a bench calculator to do that we will start bc and 4567 modulo 256 and you get the answer 215 and I can come out of the bench characteristics by pressing control d. So, either come out. So, this is an illustration to show that you can actually have any arbitrary exit court except that the number has to be between 0 to 155 and therefore you would have the modulo of 256 being taken for any arbitrary exit code that you may want to throw from the programming.

And we have illustrated this using bash shell program itself. Why do we need to learn the exit codes. This is because you have different situations while running multiple processes. Sometimes you just want to run the processes one and two after each other without worrying about the exit course. But if there is any dependency then you may have a situation where you want to run the first process and went to successful only then you want to run the second one or you may have a situation when you have the first process and if you did not succeed as a plan b you may want to run the second process.

So, you can combine two different processes on a single command line while also taking into account the exit codes of those processes. And you can combine these not just with a pair of processes but many, many more on a single line of command. We will look at that in detail in the following session.
(Video End: 31:50)