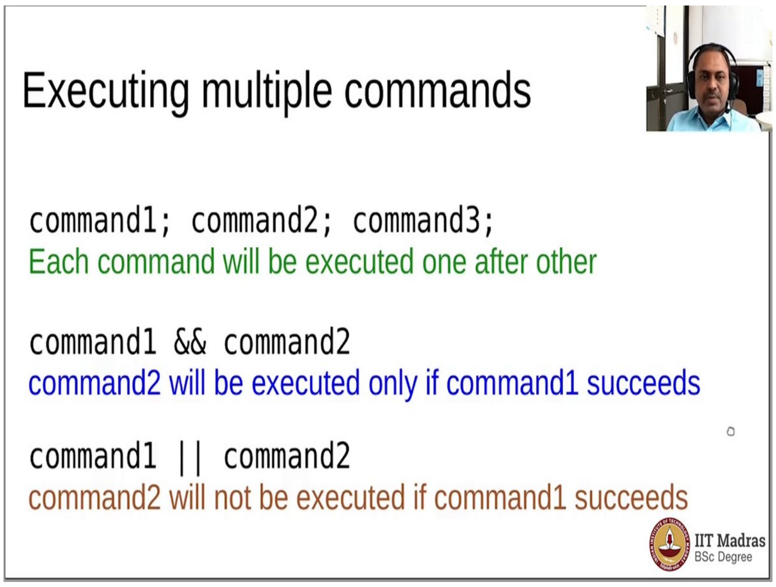
System Commands
Online Degree Programme
B. Sc in Programming and Data Science
Diploma Level
Prof. Gandham Phanikumar
Department of Materials Engineering
Indian Institute of Technology- Madras

Lecture 7
Combining commands and files

Welcome to the session on combining commands and files. Linux is a modular operating system one can combine different commands to achieve a complex task. One can also use multiple files for both input and output in this session we will see how we can combine both commands and files. So, that a single command line could actually achieve a fairly complex task.

(Refer Slide Time: 00:41)



Executing multiple commands

```
command1; command2; command3;
```
Each command will be executed one after other

```
command1 && command2
```
command2 will be executed only if command1 succeeds

```
command1 || command2
```
command2 will not be executed if command1 succeeds

IIT Madras
BSc Degree

So, here are few examples of how commands can be combined on the command line the very first way is something that we have already seen you have the commands that are executed one after other irrespective of whether the commands have been successful or not. Now the second means is where you have got the two ampersand symbols which would lead to an effect where the second command is executed only if the first command is successful.

There is one more way of combining the commands where you have got the two pipe symbols between the two commands the second command would be executed only if the first command were to be false. The written codes for each command something that we have discussed in the previous session would be very useful because if the written code was 0 then it would be true and if it were to be any other value greater than 0 then it would be false.

We could use the Boolean values indicated by the written code to combine the commands appropriately we could see that the double ampersand as used for the and logical operator and the double pipe that is used as our logical operator could be then interpreted similar to the Boolean variables combined in the same manner.
(Video Start: 01:58)

We have seen in a previous session that we can combine multiple commands on the command line. So, let us combine them for example ls and then date command and then counting the number of lines in a file called etc profile. So, these three commands are then executed irrespective of whether they are successful or not. So, let us make one of them fail. So, we would type as a ls slash blah which we know that does not exist and then edit command and then wc -l etc profile.

And you see that the first command has failed nevertheless the second and the third commands have gone on to execute. Now this is a behaviour that we could actually change where the execution of the second or third commands can depend on whether the first command has been successful or not. So, we will come to that in a moment but before that I want to show you that there is one more way by which you can execute three commands on the command line together as a list of tasks.

So, I would start them with a parenthesis the bracket you could see that the same action has been repeated. So, there is no difference however when you use the parenthesis what actually happens is that the command is executed in a subshell which is returned back to the shell that we are using we can inspect the variable shell variable called bash subshell. So, this bash subshell is a variable that contains an integer which tells you at what level of execution we are.

So, let us try that out by passing it on to the parenthesis to see whether the number remains same or not. So, I would now say echo. Now you see that the number has changed which means that this particular command has been executed in a subshell and then is returned back. So, we could then use this to check how the execution is happening when we pass multiple commands and you could see that now the ls and date were executed in a sub shell and the value of a bash sub shell is then 1.

So, you could actually use multiple parentheses to have subshells within subshells and we can try that out right away. Now you could actually create a sub shell within this by editing it like this. Now you see that the bash subshell is reporting a value of 2 which means that from the bash that we are actually in a subshell has been created in which the ls command has been launched and then another sub shell has been created within which the date command has been launched.

So, we could verify that as follows. So, you would see that the first command ls is executed here and then the bash sub shell shows that it is number one and then another subshell has been spawned and the date command has been executed and then the bash subshell value is two which means that the date command is executed in a shell launched two levels below what we are actually running.

Now why is it useful? Sometimes this is useful to ensure that the environment within which the execution is happening is closed it is not a good idea to launch sub shells indefinitely because each time a subshell is launched the environment has to be prepared for that and therefore it could be expensive computationally. However it has certain usages where you could provide a certain environment namely certain values of the shell variables that are created for a particular executable and so on.

To be provided so that the execution can happen in a subshell without being interfered by other variables that may be present in the parent shell. Now let us see if we can run the commands where the execution depends upon the exit status of one of the commands. So, we have run two commands one after other and we will. Now make the first command fail and then we will actually have ampersand twice to indicate and operator and then we have the next command date that is being executed.

Now you will see that only the error message for the ls command has been displayed but the date command did not execute what this means is that the error code returned by the ls command is false which means it is more than 0 and therefore the second command namely the date command did not execute. Why is this useful? Because it helps us in ensuring that the second task which depends upon the first task will execute only if the first one is successful.

Now let us look at a possibility that is exactly the opposite. So, here we are combining the commands where we are using the double pipe which is for the r operator and you see that the date command has executed though the first command has been a failure. So, this is a situation where the second command is like a plan b where if the first command failed only then the second command would execute.

I f the first command were to be successful the second one will not be executed and that is something that we can see right away and you see that the ls command was successful in displaying the list of files and directories in the current directory and because it was successful the second command date did not execute. So, here is a illustration where the three symbols the semicolon the double ampersand and the double pipe could be used to combine commands in such a way that they can be executed in three different means.

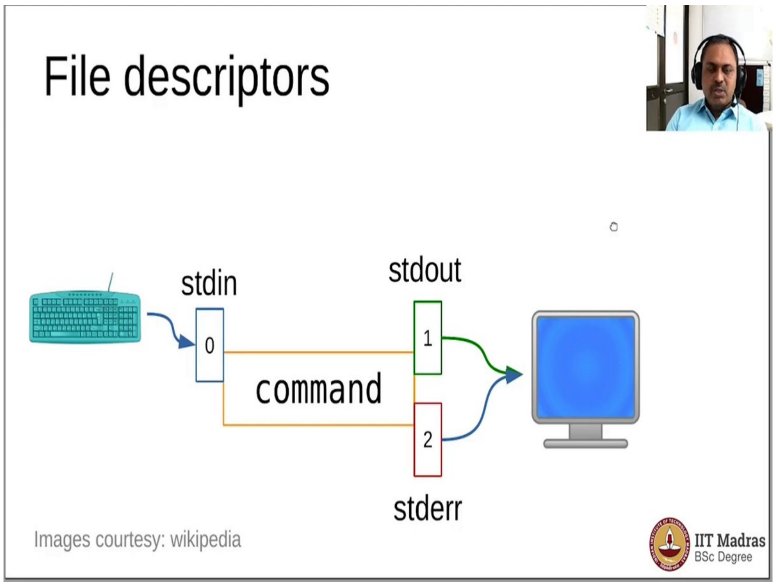
That is independent of the execution status dependent on the execution status to be true and dependent on the execution status to be false. So, now it is up to our imagination to combine these in very many ways by which we can actually achieve a complex task that depends upon the commands that are executed before that let us try that out once. So, here I am running three commands and I am using the and operator for all the three.

So, if any of them fail then the next set of commands on this command line will not get executed. So, we will make the first one fail that is quite straightforward we just need to try to list commands of a non-existing directory therefore nothing after the ls command will get executed. Now we could actually make the first command successful but the second one failed by giving some arbitrary option which we know that will not exist and you see that the date command has been given an option -Q there is no such option.

So, the date command has failed and therefore after the date command the wc command did not execute however the ls command went on to execute successfully and that is why the date command was tried out. So, like this you can actually combine multiple commands using both ampersands as well as pipes to achieve the Boolean logic that you would desire for the total command that you want to execute which could be achieving a fairly complex task.

Here we are using silly examples like ls or date but it could actually be a bash scripts that you would have written which actually takes at an action on your data or processing of some information that you have or perform network operations and so on. So, each of these commands can as well be fairly complicated bash commands and the logic of combining them on the command line would remain the same.

(Refer Slide Time: 10:11)



File descriptors

Now we come to the concept of file descriptors. So, in C language when you learnt how to read and write content with the files then you would have come across the concept of file pointers where you have a file pointer pointing to a file which you could read from or write to. So, every command in Linux has three file descriptors they are called as the standard in standard out standard error.

So, the standard in is a pointer to the stream which is coming from the keyboard or the user input and the standard out and standard err are usually pointing to the screen where the display of the output is made numerical values like 0 1 2 are assigned to the file descriptors standard in standard out and standard err respectively. So, that you could actually use these numerical values on the command line for redirecting some of those outputs are inputs to either a file or a command.

Now in Linux we should imagine that each of these three file pointers are actually looking at only the stream of characters. Now that we could understand that there are these three file pointers. The next thing that Linux provides us is that we can redirect these three streams to either a file or a command or the default behaviour can also be left as it is and that opens up a lot of possibilities for us to imagine how the commands can be combined.

And there are certain operators that are used to combine the commands and we will explore them shortly. So, here is the first operator that we are going to use to combine a command and a file that is the greater than symbol or the right angle bracket. So, what does it mean when we say when we write a command like command greater than file1 what happens is that the standard output is redirected to a file called file1.

And the rest of the pointers will remain the same which means that the input would continue to come from the keyboard and if there is any error that has come out of the running the command that would come on to the screen only the output is actually redirected to the file. Now the file will be created afresh in the directory where the command is being executed which means that you must have write permissions to the directory and you should not have any file by the same name already because it would be overwritten when you run this command.

So, let us see this command in execution and understand how does it work. When you have a command that has a lot of output and we would like to look at that output slowly this feature of greater than symbol to redirect the output to a file would be very useful. Now I am in the home directory and we would actually look at the listing of the user bin directory. So, ls user bin and you see that a number of files have come out of the screen and I can actually do the same thing by typing -1 user bin it would give one command per line of the output and we would like to then write this output to a file.

So, that we could look at that file slowly so that is where the greater than symbol comes and then I have to give a file name. So, I would give my file name as file1 and then the file1 has been created and you can see that there is no output on the screen. The reason there is no output on the screen is because there is no error that has actually come when I executed this command.

Now I look at the contents of my home directory and sure enough there is a file called file1 that has been created and it has 26 kilobytes let us say quite a bit of a content because there are lot of files in the user bin directory of my computer and we can now inspect the contents of file1 using the less command. So, you can see that the list of file names are being displayed on the screen.

And I can scroll through at my leisure I could also go back with up arrow key and look at them and I can press space bar to scroll page by page and when I am done I can press q and come out. So, you can see that we have the ability to pass on the output of a command onto the file. Now if I happen to make an any error and there is no output the file will be empty.

So, now you can see that the file1 is 26 kilobytes. Now what I would do is I would actually intentionally make an error. So, I would try to list the contents of a file called slash blah which does not exist and I rewrite the output to file1. Now what happens is that there is an error and the error has actually come out to the screen and we can inspect the contents of the file1 and you see that it is zero that means that there was no output naturally because there is nothing to report as list of contents for this slash directory which does not exist.

So, you can also see that a file which was actually having 26 kilobytes has been overwritten and it is now an empty file which means that when you use the greater than symbol you must be aware that any existing file will be overwritten and therefore you should use fairly unique file names when you are in the beginning stages. So, that the danger of overwriting does not hurt you in losing any of the files that may be valuable to you.

So, this is what we have seen the command is ls and the file1 is a text file and when we ran it what happened is the output was actually put in the file and any error was actually put in the screen for us to view and the command did not have any input and therefore the standard in was not very relevant. Now we will put the greater than symbol usage to some use where we can actually look at the output of a command which scrolls a bit too fast.

So, let us do that with a command called hwinfo and when you type that it is inspecting the various hardware and it is now dumping the output onto the screen. So, we then write the same command greater than and I need to give a file name because this is useful information I would like to type it as hwinfo dot txt to indicate that it is a text file and it has the information coming from the command hwinfo and you could see that there is nothing on the screen.

Now because everything has been put onto the file, now let me look at the file by using the less command and you see that it is giving me information about the system and you see that it is telling me about the kernel that was booted and initially how the probing of the hardware is happening and as you scroll through you will see information about your computer hardware and we will look at that in the next session to inspect the computer hardware.

But for now you could actually see that the content can be made available for you to read at your leisure. Now if you try this command in a folder where you do not have a write permission you will get an error and let us just check that out also. So, we would go to let us say slash user and we would actually look at the permissions whether we have write permissions or not and you see that the current directory has write permission only for the root and I am not root right now.

Who am I? I am another user and therefore I do not have a write permission because the file is owned by root and only the owner has write permission and I belong to the other category and I do not have the right permission. So, here if I try to run the command then I will get an error. So, let us say I type ls greater than and then file1 and you see that permission denied is the error that I get it is good because I do not unintentionally make any changes to the system folders.

And this is a security measure that is built in and that is one of the reasons why we should not be logged in as a root user with administrative privileges when we are a novice user unknowingly sometimes we may make some errors. So, before you run the command you always ensure where are you and when you are in your home directory then you can do these commands without any error. Now the greater than symbol can also be used to actually create files.

So, let us look at the help for the cat command and it says concatenate files and print on the standard output and which actually can be interpreted with respect to the information that we have about a command. So, we have cat and then greater than and then a file okay. So, what happens when we type this command is that the cat command is supposed to receive the input from a file that is listed on the command line.

But we did not specify any file if you did not specify a file on the command line for the cat command to read the content then it would read that content from the keyboard which means it is reading from the standard in and we can then see how we can type some command which will then go into the file called file1 because the greater than symbol indicates that whatever cat is supposed to display will be put into the file1.

So, you see that the command prompt is now empty and it is expecting us to type some we can type some content now. So, that is my first line second line goes like this and how do I come out you could come out using control plus d option okay. So, I do that. Now and I have come out now what happened is that the three lines I have typed have been read by the command cat and they are supposed to be displayed onto the screen.

But because of the greater than symbol they are being redirected to file1. So, let us look at the contents of the file1 you see that the file1 has 170 bytes it is a short file and as we have already mentioned earlier it has been overwritten by this command because earlier it had some other content. So, we have overwritten it. So, that is not a problem because these are all only meant for practice.

But remember always that whenever you use greater than symbol that file will be created afresh and if a file exists by that name it will be overwritten. Let us look at the contents of the file file1 and you see that what we have typed is there now on the screen for us to see. Now you can see that the difference between cat greater than file1 and a cat file1. So, it actually what it does is that greater than actually would take input from the keyboard and create a new file by the name file1 whereas without the greater than symbol the content of the file 1 will be displayed onto the screen.

So, it is very important for us to understand how the file pointers are actually being redirected. So, here you can see that by default the cad command was taking input from the standard in keyboard and it was writing it out onto the screen but we have put a greater than symbol. So, it was writing on to the file1. Now let us test our understanding of the input output redirection by seeing what happens when we just say type cat.

What happens is that there is no file specified on the command line which means that the input for the cat command should come from the keyboard and the output should also be on the screen. So, which means that the command would now take input from the keyboard and place the output on to the screen and let us see how that happens. So, as soon as I press enter you see that that particular line has been read and it is displayed onto the screen.

So, you can see that this command is actually processing the text line by line and you can actually finish up this task by actually pressing ctrl D, so that you can come out of the command. So, finish up by end of the file using control +D to close. So, now I press ctrl D and what happens is that it has already finished writing the content on the screen. So, there is nothing that has happened.

So, this actually helps you understand how the file pointers are being interpreted. So, there are three file pointers standard in standard out and standard err and we have to be very conscious which file pointer is pointing to which of the aspects namely the keyboard or the screen or the file you could actually use two symbols of greater than or the right angle bracket.

So, if you use these two symbols side by side without any space between them the interpretation is that the output of the command should be appended to the file1 by appended we mean if the file1 exists then the content will be written to the bottom of the file and if the file1 does not exist then it will be created and it will be written from the top. So, it means that it is a bit it is a slightly safer option to use.

So, that if there was any file then it would not be overwritten it will only be appended to okay. So, that is why here in the graphic I have shown the arrow pointing to the bottom of the file1 not to the top and that is a hint that the content will be appended to the file1 and not replaced the warning of the file1 to be created remains and you will have errors if you run this command in a directory where you do not have permissions to create the file1 because you do not own that particular directory.

Now let us try out the double greater than symbol to see how to append the content of any file by using the commands and the output would be written to that file in an appending mode. So, let us try that out. So, I have this file1 and file two I will just erase all of those okay. Now I do not have any files. So, I now use the command to also show you how it will be created when it does not exist.

So, what we do is we type date and double greater than and we'll say file1 and you see that a file has been created. Now afresh though we have used the double greater than symbol because if the file was not existing it will be created and if it was available then it will be appended to. So, I run the command again. Now and once more and once more I have run totally four times.

So, let us look at the contents of that particular file and you see that there are four lines and from the seconds you can make out that it is actually appending one after other. So, this way we have not lost the content that was appended or inserted earlier and therefore this mode is preferable when you want to preserve the contents of the file while redirecting the output of a command to a particular file that you want to inspect later.

So, when you want to write multiple commands where the output would be redirected to the file and appended later on with another command then you could combine all of them also in the same line using the semicolon as follows. So, you would type date and let us say file two and then we have another command let us say wc -l and then a third command perhaps to inspect the file user bin znew and that output also I would like to write to file2.

So, you can see that I am combining three commands and the output of each of those is being written to the file and then I can actually look at the content of the file too and you could see that the three outputs have been appended and we are seeing all of them. Now what I would do is I would make an intentional mistake here the second one I would actually make with only one greater than symbol and you. Now see what would happen you see that only the second and third outputs are there.

The output of the first date command is gone because the second command has actually overwritten it because it has created the file to affect okay. So, this is how you should interpret the greater than symbols. So, double greater than and the greater than both will actually create the file if it was not existing but double greater than would actually append if the file was already existing and therefore the safer option for you to explore.

Now we will use the same double greater than symbol to append text to a file that we are creating. So, we would remove these files. Now we would actually try to append to a file and the file does not exist initially it does not matter. So, I will have file1 and I want to write something here. So, I would write like this okay first attempt to add text to the file file1. Now I come out using control +D. Now I have come out. So, I would run this command once more.

Now I say this is my second attempt to append text to the file called file1. We can inspect the content using cad or less or more commands okay I come out using control + D. So, we have done two attempts. So, we would do one more this is the third attempt to append content to the file called file1. So, we have. Now done three attempts and let us see what is the content of the file1 and you see that all the lines are there all the six lines are there the first attempt had these three lines which are now reproduced here.

The second attempt had these three lines and they are here now and the third attempt had only one line and that line is here. So, you can see that you could use the cat command to append text to your file. So, you can actually create files with the text that you want to create in it by using the cat command the double greater than symbol to append and this is one very simple way by which you can create text files.

We will learn how to edit these files in a visual environment within the command line in one of the sessions later on but this is a very quick and dirty way by which you can actually create text files and append stuff to that one by one.
(Video End: 31:32)