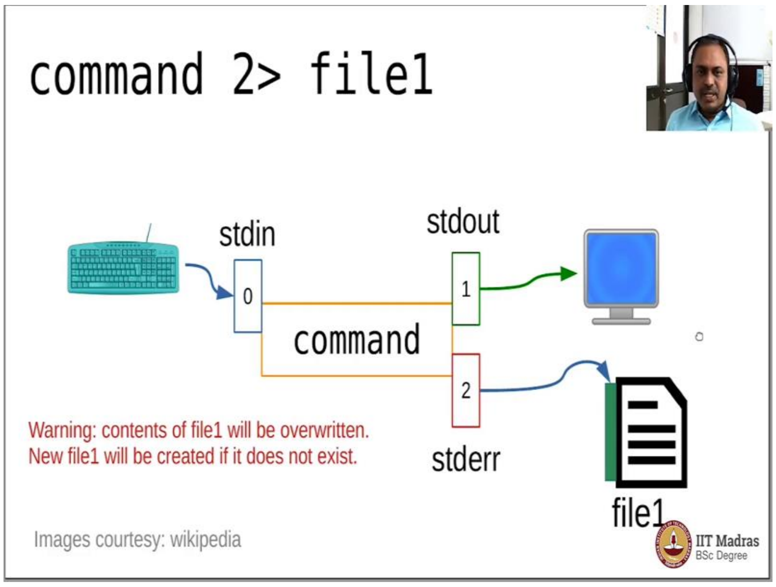
System Commands
Online Degree Programme
B. Sc in Programming and Data Science
Diploma Level

Prof. Gandham Phanikumar
Department of Materials Engineering
Indian Institute of Technology- Madras

Lecture 8
Redirections

(Refer Slide Time: 00:15)



```
command 2> file1
```

Warning: contents of file1 will be overwritten.
New file1 will be created if it does not exist.

So, here is yet another operator which is actually a 2 greater than command space 2 greater than space file2 greater than is an operator with which you should not insert another space between them. And what this operator does is to redirect the standard error to the file1 the standard output would be having its default behaviour namely send it to the screen the standard end will have its default behavior by reading it from the keyboard.

And therefore the only one file pointer has been rejected which is basically the standard error to the file1. Now why is it useful? We will demonstrate the use of this particular command in a moment again the same warning applies that is if the file1 does not exist then it will be created. So, you should watch out what is the name of the file this territory is going to be created whether you have permission for that whether there is any other useful file already by that name which you do not want to overwrite and so on.

So, if you look at this command ls and the dollar home you will see that the list of files and folders in the home folder are being displayed on the screen and you can give multiple options to the ls command. So, I would say ls dollar home and then let us say slash blah. So, what happens is that the territory dollar home has content and that is being displayed on the screen.

But the directory slash blah does not exist and therefore there is an error message on the screen. Now both error messages as well as the output have come onto the screen and our objective of using 2 greater than operator is to return to the output onto the screen but the error onto a file. So, we will do that now. So, I use up key to retrieve the command back for editing and I press 2 greater than and here I need to give a file name where the error will be placed and I would error dot text.

Now if you see the screen output is very clean because it does not have any error message and there is a file called error dot text that has been created and you can actually look at its content. So, it has 53 bytes and cat error text would actually show you the error that was displayed when it was trying to run the command ls on the directory called slash blah which did not exit.

Now what is the use of this 2 greater than operator? It is 2 purposes one is to keep the output on the screen clean. So, that the error messages are not being displayed for whatever reason another way is actually to send those error messages to a file. So, that you want to inspect that file and understand what errors have come up that you want to debug at a later point of time you can combine both greater than as well as 2 greater than symbols.

So, that the standard out is related to a file and the standard error is related to another files. So, you could actually try this out. So, that the output is in one file and the errors are all in another file and this is a very useful feature because the error file will be requiring our attention to watch out if there was any problem with a our script whereas the output file is something that we may want to keep for our log.

Naturally the warning about the file1 and file2 being created a fresh will stay that is the directory should have the right permissions for us and if there are any files by those names. So, they will be overwritten and therefore we have to watch out what names we are giving for those files.

(Video Start: 04:01)

So, now we will try to do this redirection as described just a moment back. So, we have a ls dollar home and slash blah and then we were at greater than let us say output that text 2 greater than error text. Now if I press enter what do you expect to come onto the screen both the standard output and standard error have been redirected you do not expect anything to be shown on the screen and that is exactly what happens.

And here is again another application for you to imagine that if you do not want any output on the screen and the script has to do its job silently by writing stuff onto files rather than onto the screen and this is one way by which you can achieve that. Now let us look at the 2 files that have been created. So, the error file has been created 53 bytes and the output file has also been created 126 bytes.

So, what are the contents? So, the error file naturally has the content which is expected because slash blah does not exist there is an error message for that and the output file will have the list of files and directories in the current folder of dollar home which is the home directory. So, that is also as expected. So, as you can see we can actually combine both of these in a very useful manner.

So, what happens is when you try to look at the ls output in a recursive manner usually there will be some errors on the screen if there are any directories which are not supposed to enter because they belong to the system. Now there are such directors available in slash etc folder. So, let us try that out. So, ls -r slash etc -R is 2 recursively traverses that particularly directory and just list those files onto the screen.

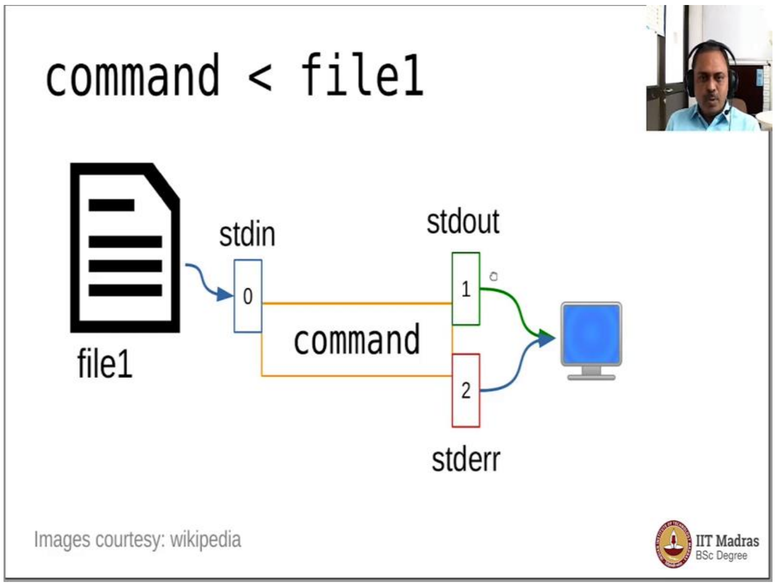
So, when you type this what happens is that it is running okay and we can look at the contents of the folder slash etc in a recursive manner and whatever are the directories and their listing we can put that in the file called output dot text and if there are any error messages because we are not supposed to enter some of the directories then that would be made available in the text file error dot text.

So, when I enter here I should not see anything of the screen as expected and let us look at the content of these 2 files. So, you can see that the output file has 54950 and fifty bytes and the error file has 327 bytes as you can see that the 2 files have been overwritten from our previous command. So, we have to always watch out that the greater than symbol and 2 greater than symbol will always overrate the files if they are already existing in the current folder.

Now let us inspect the content. So, I would use the ls command to inspect the content of output and you can see that the files in the etc folder are being displayed here and for every directory it would actually list the name of the directory protect and then the files that are in that particular folder are listed. And it would inspect every folder recursively and this is a very nice way by which we can actually look at what all the files that are there in the particular directory in a recursive manner.

But in the process of doing that there are some directory which you are not supposed to enter and so we got some errors and let us look at that it says that there are some directories which are not supposed to be entered by us and therefore we got permission hidden errors and those errors are captured in the error text. We can use the less than symbol or the left arrow bracket to indicate that the standard input has to be redirected.

(Refer Slide Time: 08:00)



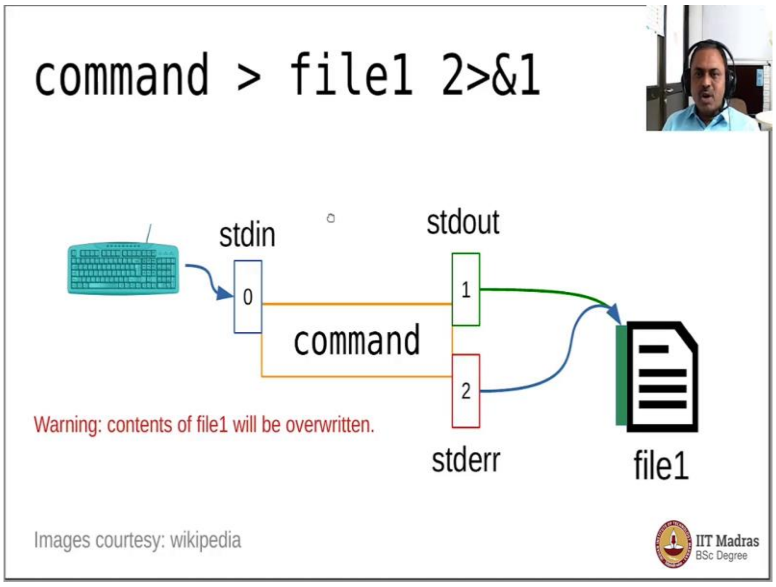
So, any command that was expecting the input from the keyboard could actually take the input also from a file if we use this operator and if we can try that out with some commands that we have just been familiarized such as cat. So, let us try that out. So, we have some we have some files here. So, error text is one file that we can try out. So, when we type a cat error text cat error text then it should just display the contents of the file.

Now we can actually say cat less than error text and what would happen is that it would read the content from error text in display. So, the behaviour is exactly the same. Now we can test less than symbol by using the wc command to count the number of words lines etcetera. So, we have already seen that in the wc command is similar to us. So, normally the wc command is run on a file.

So, we have seen that we were running it on etc profile and there are 27 lines of 97 words and 581 characters in that particular file and if you were to run it on the error text which was created just now. So, it has 5 lines 35 words and 327 characters and look at the contents and you could actually make out that these are the 5 lines. Now what happens when we type like this wc and then less than.

So, if you type like this wc less than error the text it would mean that the content has to be read from the file and that would be the same behaviour as what you have given on the command line.

(Refer Slide Time: 09:45)



command > file1 2>&1

Warning: contents of file1 will be overwritten.

Images courtesy: wikipedia

Here is a very interesting combination of the operators. So, here you can see that the greater than operator is telling that the output from the command is being redirected to the file and the 2 greater than is indicating the standard error and that is being redirected to the first stream which is standard out. So, what is happening that the standard out also is going to the same file. 2 greater than indicates the standard error and the ampersand and one indicate that it should be related to the file 0.1 which is standard out it means that the standard error will also go to the same file, file1.

So, here is a very unique situation where the outputs from the commands as well as the error from the command are both sent to the same destination file which is file1. The usual warning about the file1 being overwritten and the permissions to create that file should be provided etcetera will apply. Now let us look at the application of the command that we have just now illustrated. So, we have got the example here, so ls dollar home slash blah.

So, we saw that this command would have 2 outputs one is the standard out which would be the directory contents of the home folder and the other output is to standard error which is basically the error that the folder slash blah is not available. Now we would like to combine both of them to a file. Now what happens when we do like this? So what happens is that the output alone is sent to the file1 but the error is not sent and that is coming onto the screen.

Now what happens when we do like this? So, we have sent the output to the file1 the errors also to the file1 but does file1 contain both. So, let us look at and you see that the file1 does not contain both it contains all without the reason is that the error actually has come out first and it was written to the file1 which is then over return when the output is being related to the file1.

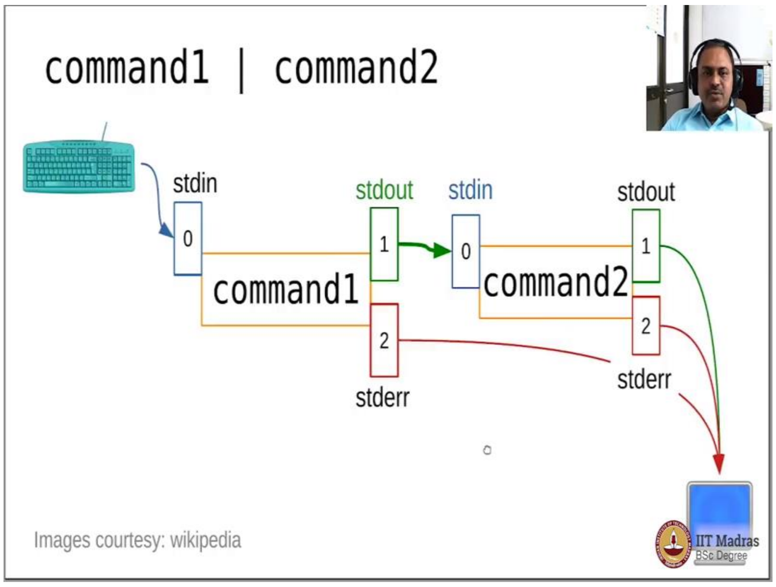
So, if you want to have both the error as well as the standard out to be related to a file then you cannot actually use this particular format because the file created by the error redirection would be overwritten by the output redirection. So, for that what we need to do is to redirect the error onto the out it would have no overwriting happening. So, a better way is to redirect standard error onto the standard out and then let this standard out be redirected to a file.

And thereby there will be no loss of the content that is being stored as file from both the streams. So, we will try that out now. So, that is a command and as you expect nothing should be on the screen because the error as well as the output are both redirected to the file1. So, nothing is displayed on the screen and let us look at the contents of the file1 and you could see that the error messages as well as the directory listing of the home folder are both in the same file file1.

So, this is a very useful feature where output from the command both the standard error as well as the standard out are both redirected to a file which you may want to look at later on to debug if the command happens to be fairly complex. In this case it is a simple command but you could actually have a shell script in the place of the ls command and this kind of interaction could be very useful in debugging what is happening to the script.

So, just to recap we redirecting the standard error onto the standard out which is then reiterated to the file1 and therefore file1 would contain both the outputs from the 2 streams named the standard out as well as the standard error.

(Refer Slide Time: 13:55)



Now comes a very powerful operator called the pipe where the output of the command 1 is sent to the command 2 as an input which means what you are doing by this pipe is the standard out is being mapped onto the standard in of the command 2. If command 2 was expecting any input from the keyboard or from a file then it could actually take it also from the output of the command 1 and thereby you could actually combine the 2 commands in order to process the output of the first command as an input to the command 2.

Now this is a very very popular way of using the commands because it allows us to combine multiple commands for processing without writing to a file intermediately. Now we have not talked about the standard error by default the standard error would actually go to the screen and that is how we are actually showing it in the graphic. So, let us illustrate the use of pipe command.

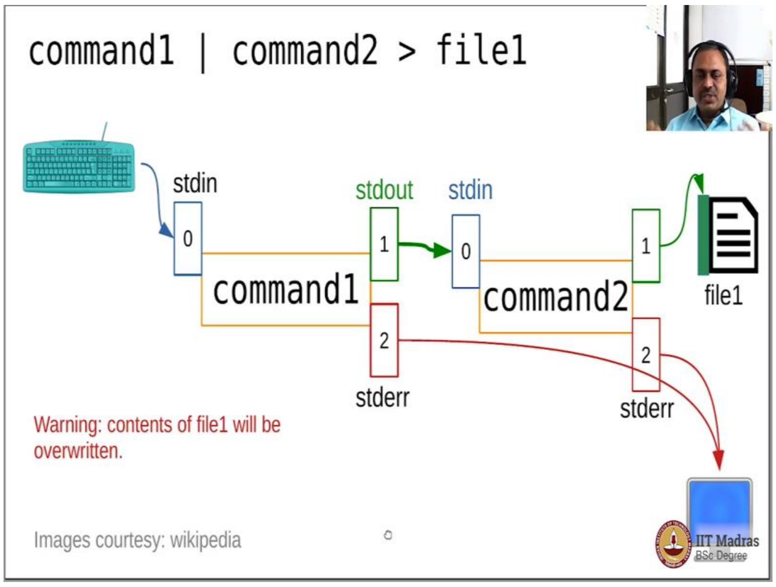
So, let us say we would like to count the number of files in the directory user bin one way is to redirect the output of ls command onto a file and then count the number of lines in that file. So, we have redirected the output of the ls command onto the file1 and we are. Now running wc on file1 to count the number of lines and we see that the number of lines is 2596.

Let us look at the file1 itself it contains the list of files that are there in the directory called user bin. Now here what we have done is to write the output to a file and then look at the number of lines in that file this is not very efficient because we are actually writing a file and then again reading it. So, we could actually do it in the memory where the output of the ls command is mapped as an input to the wc command. So, that is exactly what the pipe does.

Now you see that the operation is much more efficient and we are getting our answers straight in one go and there is no file1 that is created. So, this pipe command can be used to combine multiple commands and these is one of the most popular ways of combining commands on the shell on the batch and with often complicated commands involving several pipes symbols on the single command line are very much possible.

Let us say I would like to look at the files that are in the user bin folder but I would like to scroll through them and I do not want to write to a file. So, I could also do that now here. So, ls user bin and I can pipe it to the command called less and you could see that it is now available for us to scroll through up and down and when we are done we can press q and come out the list of commands in the user bin directory are now not written to a file but send to the less command by a mapping of the standard output of the ls command to the standard in of the less command. Thereby the mapping is in the memory and therefore it is very efficient.

(Refer Slide Time: 17:24)



```
command1 | command2 > file1
```

Now, we can actually combine both the pipe as well as the greater than symbol to combine multiple commands as well as files as destinations for the standard output. So, here is again a very popular usage where the command 1 is actually combined with the command 2 using a pipe symbol which means the standard output of this command 1 is mapped over to the standard in of the command 2.

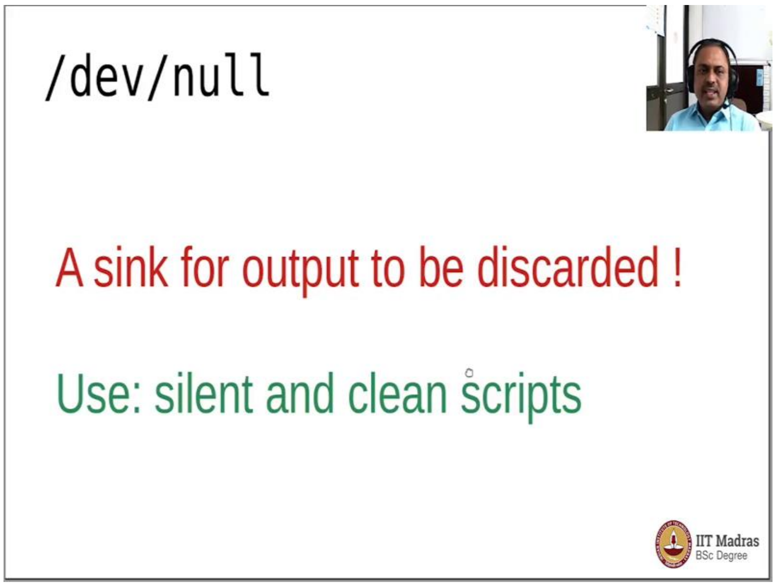
And the standard out of the command 2 is then related to the file1 however the standard error of both the commands is still shown on the screen by default because we have not done anything with those 2 file pointers and now naturally the file1 has to be created afresh because the greater than symbol would actually mean that it has to be created afresh and so if that file exists earlier then it will be over written and so watch out for that.

We can try out an example to illustrate how this can be done. So, have a look at the following commands that we are going to do a demo. So, let us say we would like to list the number of commands that are there in the user bin folder and write that output onto a file so that we could actually use that file for some other purpose. So, if you were to do like this ls user bill pipe wc -l that means that the number of files in the user bin directory are being listed and greater than and then file1.

So, what would happen is that file1 would have the number stored which means that we have done 2 actions here one action is the standard output of ls is passed on at the standard input for wc and the standard output of w c is reiterated to the file called file1.

The file1 has the content which is basically the number of lines that has been counted by wc which is the number of files in the directory user bin.

(Refer Slide Time: 19:10)



/dev/null

A sink for output to be discarded !

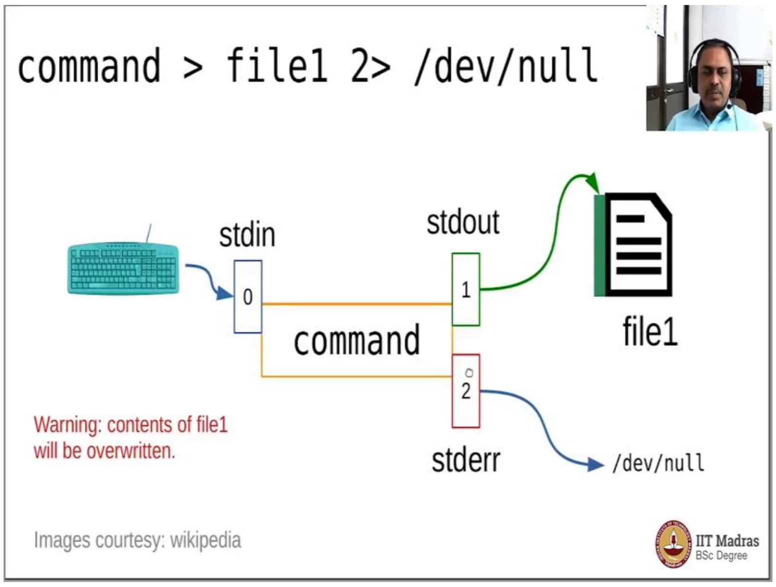
Use: silent and clean scripts

IIT Madras
BSc Degree

There is a file in the slash dev directory normally we do not go there to fiddle around with those files because they belong to the system and some of them are very sensitive. So, we do not normally do anything with the slash dev folder but there is one special file their called devil which is like a black hole. So, if you happen to write anything to that particular file then that content is basically disappeared from the system and it is actually used as sync.

So, sometimes when you have confidence that the script is running well you do not want to have any error from that script to be displayed on the screen for the user then you can actually read it the standard error of the command to dev null and therefore the errors will actually get disappeared into the slash dev slash null folder and thereby they will not be seen on the screen and therefore the shell can actually have every clean output on the screen.

(Refer Slide Time: 20:08)



```
command > file1 2> /dev/null
```

stdin
0
stdout
1
command
file1
2
stderr
/dev/null
Warning: contents of file1 will be overwritten.
Images courtesy: wikipedia
IIT Madras BSc Degree

So, here is a typical usage. So, you have got the command that is running it is giving out 2 streams of uh text one is the standard out which you may want to redirect to a file but if there were any errors then you do not want to keep them you do not want to also show them on the screen. So, you can actually send it off to dev null. So, that it is disappearing and that the redirection of standard error is already seen.

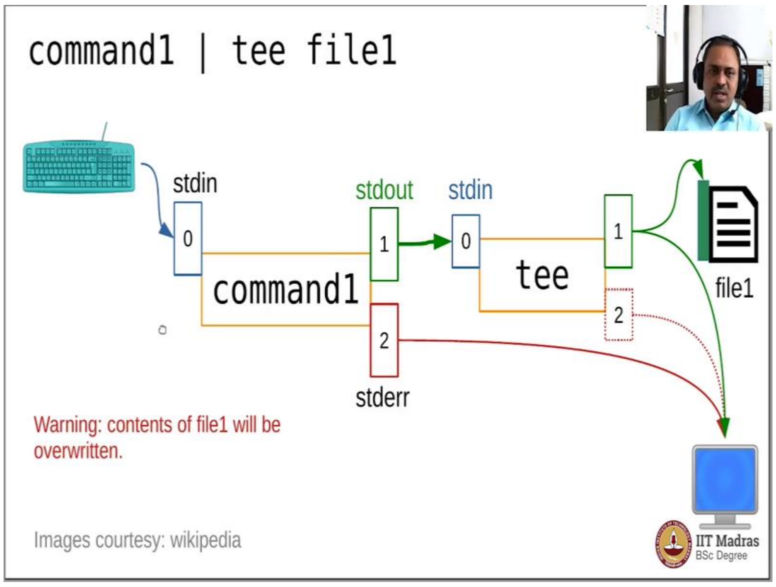
By as it is 2 greater than and the file itself is a special file where it actually destroys that content and therefore the other messages are not coming onto the screen which is the purpose of this kind of a command. Now to illustrate the combination of both greater than and dev null we have got the usual example of listing of the folders in the home folder as well as in a directory slash blah which does not exist therefore it will give you errors.

So, here you see that the errors have become onto the screen but also the contents of the dollar home. So, we would like to have only the contents stored in a file called file1 but the errors are coming onto the screen I do not want them. So, what I do is I read it the error to dev null and you see that the screen is now very clean because there is nothing on the screen the output is only in the file called file1 which we can inspect by using the cat command.

And the errors have all disappeared into the black hole called slash dev slash null. We have done this exercise with respect to the recursive listing of directories in slash etc folder where we face some permission errors to enter some directories. So, again here we can do ls -R slash etc we want to redirect the output listing to the file called file1. We would like to have the errors disappear to dev null.

And you see that the command has executed in a very clean fashion typical of Linux command and you could look at the file1 contents it would contain the list of files in the etc folder recursively and the errors have all gone to definitely. So, that is a very clean behaviour of this command and this is a very typical usage of the 2 symbols greater than as well as her 2 greater than.

(Refer Slide Time: 22:36)



```
command1 | tee file1
```

Warning: contents of file1 will be overwritten.

So, here is yet another command which helps us in organizing the file streams in a very clever fashion the tee command what it does is like the t shape what it does is that it would actually split the standard output into 2 streams one stream is redirected to the file1 and another copy of the screen is then sent on to the screen. So, this is used in situations where you would like to have a copy of the output stored in a file but also sent to the screen.

So, which means that you could then further do another processing by writing something else after that tee command? So, we can see how it is used by some illustration. Let us test the tee command once here. So, we look at the help for the tee command what it says is that it would read the input from the standard input and write it to both standard output and any file that has been specified okay and it is using plural.

So, it could actually write to multiple files okay and because it is reading from the standard input you could actually put the tee command after a pipe. So, that the output of one command is sent as an input to the tee command which it would actually put it on the screen and also write into file. So, we will tie that out using our favourite command a simple command ls dollar home and pipe tee and then we write a file call file1.

So, you see that the contents of the file1 would have the same thing what is on the screen. So, you can see that the output is the same which means that the tee command has a placed a copy output of ls command in the file called file1 and it is also sent it off to the screen for you to see. What happens when we give 2 file names. So, you could see that cat file1 cat file.

So, it has written to both the files which means that the tee command is useful in sending multiple copies of the standard input whatever input is coming in it would actually create multiple copies one to the screen by default and another one to the file1 another one to file2 and so on. Now what does it do to the standard error? So, that is something again we can inspect. So, here we are actually confident of raising an error.

So, because slash blah does not exist. So, when we run this command you see that the error message from the slash blah is coming onto the screen and the output is slightly different here the name of the folder is given with a colon that is typical whenever the ls command has multiple folders being listed. So, each of them will have a colon. So, that the file listing below is for the particular folder.

And the folder listing is here for the home folder and we can inspect the contents of file1 and file2. So, cat file1 it has the listing of a home folder and the cat file2 also has the listing of the home folder and look at these files. So, file1 and file2 have the same amount of data let us even compare the 2 files there is a command called diff. So, diff file1 file2 and there is no output which means that both of them are identical files. So, let us look at the help for the diff command it says the diff command does by comparison of files line by line very useful to understand.

So, what it does is the tee command has given 2 copies of the standard output on file1 and file2 both are identical and the the same content has also been put on the screen for you to see. Now you could also pass it on to another command beyond that. So, you could actually put a pipe after that and put wc –l, so counting the number of lines in the error realistic. So, now you see that the error message has come onto the screen the number of files in the home folder are counted as 14 and 2 copies of the ls command output are placed in the file1 and file2 let us verify that.

So, you could see that file1 and file2 have 2 copies and when I run the wc -l on file1 as well as on file2 you see 14 lines because that has been counted by wc but it is not counting from these 2 files but actually from the output of ls command which has been passed onto the screen by the tee command. So, you can see that the tee command helps in keeping a copy of the output in a file and also send it to another command that may actually process it further.

Because it is sending it to the standards out which can be redirected by the pipe symbol to another combine? So, that way you can actually combine multiple commands on the command line by using multiple pipe symbols as well as the tee command to keep copies of those outputs intermediately. Now we will suppress the standard error message that has come here by using the 2 greater than symbol. So, let us do that now here. So, I should do it here because here is where weather is raised. So, I would send it to the dev null.

So, now you see that the number of lines has been printed on the screen but there is no error message on the screen that is pretty neat. And we also have 2 copies of the listing of the home folder in file1 and file2 respectively. So, this is a pretty neat way of combining many, many things. So, you can see that what we have done we have given the redirection of the standard error onto the black hole called dev null.

We have pipe did the we have pipe the output to tee command which has copied the output to the 2 files and then we have piped the standard output onto the wc which has counted the number of lines. So, like that you can assemble multiple commands to achieve fairly complicated task. So, now as you can see that we are able to combine the operators as well as multiple commands and file names to achieve fairly complicated task.

Where we have one command creating an output which is then passed on to another command as an input which is then passed on to a file2 storage or to yet another command to do some more processing like counting the number of lines etcetera. So, each of these commands can actually be a batch script this is where you can start imagining that you can actually create a very complicated task to be achieved by the command line as well as multiple commands being placed in a script which we will be learning shortly.

(Video End: 29:31)