

System Commands
Professor Gandham Phanikumar
Metallurgical and Material Engineering
Indian Institute of Technology, Madras
Stream editor sed

(Refer Slide Time: 00:13)

sed



A language for processing text streams

o



Welcome to the session on sed, S E D also called a sed is a language for processing text streams.

(Refer Slide Time: 00:24)

Introduction



- It is a programming language
- **sed** is an abbreviation for **stream editor**
- It is a part of POSIX
- sed precedes awk

o



Sed is actually a programming language because you could write sed scripts and have loops. However, often sed is used on the command line as a set of commands to process the inputs text. Sed is an abbreviation of the phrase stream editor. So, it is meant to process the

incoming input text as a set of lines using commands that you are already being familiar, while going through the ed or x editor, by learning the vi editor as part of this course.

And the sed is also part of the POSIX standard. You could also instruct sed to stick to the POSIX standard features so that the scripts are compatible with other systems. As you would guess GNU offers extended features where the sed command. And sed command has come before the awk command.

So, there are things that you could do with sed which you could also do with awk and perhaps a lot more of programming aspects. But the simplicity of sed and the variety of text processing commands that are available, make it a very convenient language to do a one-line pre-processor for text processing.

(Refer Slide Time: 01:52)

Execution model



- Input stream is a set of lines
- Each line is a sequence of characters
- Two data buffers are maintained: active *pattern* space and auxiliary *hold* space
- For each line of input, an *execution cycle* is performed loading the line into the pattern space
- During each cycle, all the statements in the script are executed in the sequence for matching *address pattern* for *actions* specified with the *options* provided

○



The execution model of sed looks at the input stream as a set of lines and each line is a sequence of characters. There are two buffers that are maintained one for of the active pattern space and one for the auxiliary hold space. So, the matched patterns will then be found in the hold space if you have used parenthesis and the active pattern space will contain the line that has been read.

And the script will contain a set of commands separated by semicolons if you wish, and each execution cycle is performed on every line that has been loaded into the pattern space. And during the cycle of execution, the statements in the script are executed in the same sequence for all those commands, where the address pattern matches what is for the currently held line in the pattern space. And if it does, then the action specified will be

executed using the options that are specified after the action, the action is given by a single letter. And these letters are same as what you have learned in the ed editor or x editor.

(Refer Slide Time: 03:11)

usage

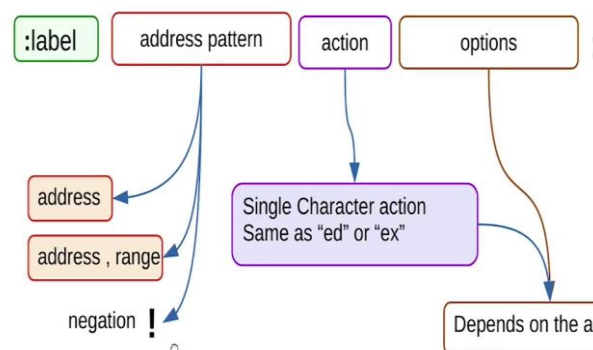
- Single line at the command line
`sed -e 's/hello/world/g' input.txt`
- Script interpreted by sed
`sed -f ./myscript.sed input.txt`



So, here is the usage of the sed command using the minus e option. And giving the command in single quotes on the command line, you could process all the lines of the input text in just one line. And the output will be returned to the standard output, so you could pipe it to another command which you may want to use for further processing. You could also place all the commands of sed in a file and give the option minus f to process these commands read from the file.

(Refer Slide Time: 03:44)

sed statements



Typical statement would look like this. At the beginning there is an optional label and then there is an address pattern. The address pattern can have the numbers, regular expressions or a range. It could also contain the ampersand symbol to indicate a negation of the address and then there is a single letter for the action that has to be taken. And then there are options which are specified as per the action and the statement can end with a semicolon in the end.

(Refer Slide Time: 04:21)

Grouping commands



```
{ cmd; cmd; }
```

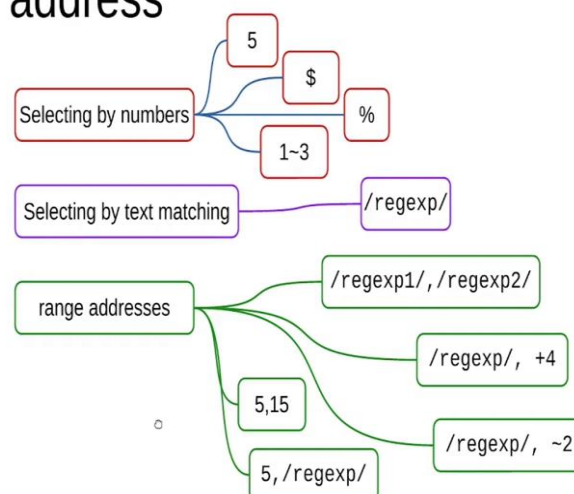
o



You could then group many such statements using the braces.

(Refer Slide Time: 04:26)

address



Now, the address specification can be given in many ways. You could select the lines by numbers. For example, if you put 5 it means the 5th line alone will be chosen for the rest of

the processing. If you put dollar, it means that last line and if you have a percentage symbol it means all the lines will be matched. And if you give a tilde, then it means that every third line starting from the first line is going to be processed if you have given 1 tilde 3. You can also set up the lines by using text matching.

So, you could use regular expression to do that. So, all those lines, which contain the regular expression match would be then chosen for the execution. You could also provide a range of addresses by giving two parts separated by a comma. So, you could separate two regular expressions by a comma. So, from the line of the first occurrence of the first regular expression until the line where the first occurrence of the second regular expression will be chosen for the commands where you have got to regular expressions separate a comma.

If you have separated a regular expression with a plus symbol, it means that the action has to be taken on the next four lines starting from the line where the first matching of the regular expression has taken place. If you have a tilde, as given here, it means that from the line where the regular expression has matched, the next line which is a multiple of two will be chosen for execution.

If you give the numbers just separated by commas, it means that you will not like to execute from the 5th line to the 15th line as an example here. And you could also give it as a number comma regular expression, so that from the 5th line till the first occurrence of the regular expression will be chosen as the lines valid for the execution of the command that you have provided.

(Refer Slide Time: 06:31)

actions

p	Print the pattern space
d	Delete the pattern space
s	Substitute using regex match s/pattern/replacement/g
=	Print current input line number, \n
#	comment
i	Insert above current line
a	Append below current line
c	Change current line



The single letter commands or actions are listed here p for print, d for delete, s for substitute. So, the options that have to be given after substitute are the same as what you have learned in the VA Editor Session. Equal to (06:51) the line number. Remember, it will also print a newline character after that.

Hash is a comment, so you could use it to include some documentation as a part of your scripts. i is for inserting the line which is above the current position, a is for appending a line below the current position and c is to change the line which is replacing the current line.

(Refer Slide Time: 07:17)

programming	
b <i>label</i>	Branch unconditionally to <i>label</i>
: <i>label</i>	Specify location of <i>label</i> for branch command
N	Add a new line to the pattern space and append next line of input into it.
q	Exit sed without processing any more commands or input lines
t <i>label</i>	Branch to <i>label</i> only if there was a successful substitution was made
T <i>label</i>	Branch to <i>label</i> only if there was no successful substitution was made
w <i>filename</i>	Write pattern space to <i>filename</i>
x	Exchange the contents of hold and pattern spaces

You could also have some programmatic aspects of sed editor. If you have specified b, it means that you are going to branch from that line to the line which contains a label. A label can be specified in a line above or below from where the branching has taken place. The label itself is specified using a colon. Capital N is a multi-line command where it would read this next line into the buffer.

q is to exit the sed without processing any more lines of input or any more commands. You could also do branching conditionally whether or not a substitution has taken place successfully, small t would work for branching when the substitution was successful, capital T would be when it is not successful.

And you could write whatever is in the pattern space to a file if you wish using the w command. And you could also swap the contents of the pattern space and hold space using

the x command. So, this provides quite a few possibilities for programmatic approach including recursion, if you are interested in processing the input for textual operations.

(Refer Slide Time: 08:36)

bash + sed



- Including sed inside shell script
- heredoc feature
- Use with other shell scripts on command line using pipe



You could combine sed along with the bash commands, because you could always use the sed inside the shell script. You could also have a multi-line command using the heredoc feature in the shell scripts. And you could also combine sed commands on the command line using the pipe symbol.

(Refer Slide Time: 08:57)



sed is available everywhere !
sed is a meant for text processing, fast in execution
use sed to pre-process input for further processing



Remember that sed is part of the Linux operating system. So, it is available everywhere. In Linux or Unix like operating systems and it is meant for text processing and it is very fast

in execution. So, consider using sed to pre-process the input for further processing by other scripts or commands that you are creating as part of your work.

(Refer Slide Time: 09:23)



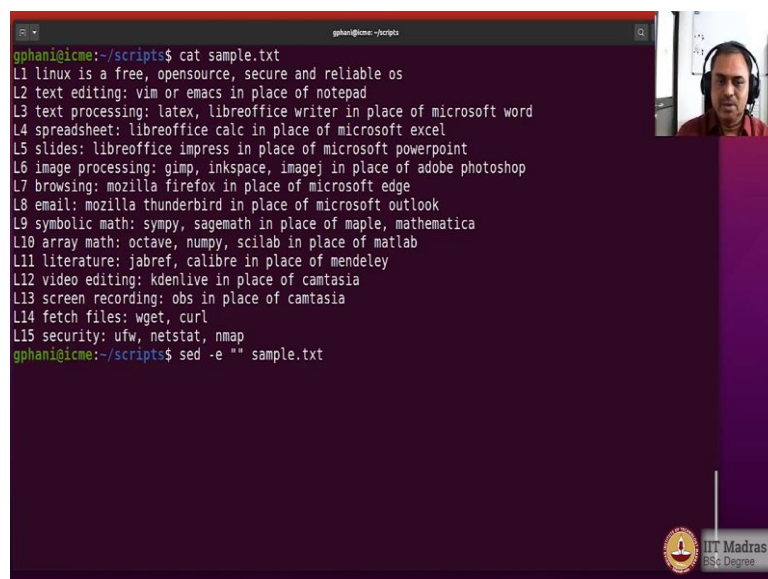
sed : stream editor

+



Welcome to the demonstration session of stream editor. In this session we would look at stream editor with the typical usages which allow for the output stream from other commands to be processed by the stream editor, so that we could use it between the pipes to process input stream in a way that is suitable for other commands that would take the process to output as their input.

(Refer Slide Time: 09:54)



```
gphani@icme:~/scripts$ cat sample.txt
L1 linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice writer in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$ sed -e "" sample.txt
```

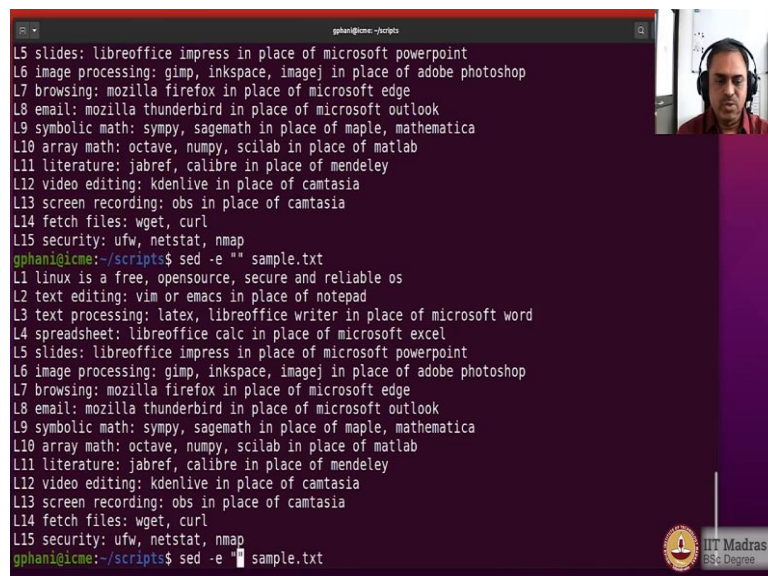
In this session, we will use typical commands of stream editor which help us in cleaning up or processing the incoming stream, so that is prepared ready for another command or shell

script for further processing. We have a sample txt file which we will use to illustrate the various commands of stream editor. So, let me show you what is there in that sample file.

So, there are 15 lines, and each line would have some sample text, I have chosen the text to be slightly useful for you when you try to read that, but nevertheless, we will treat that as any random set of text lines that we would like to process. So, the most simple usage of stream editor is actually to do nothing because the default action of stream editor is to print the line that is being sent to it.

And let us look at that command here. So, if you are giving a script that can be on the command line or in a file, so initially we would look at the command line usage of stream editor. And minus e would tell that there is a script that is coming on the command line. And I have enclosed the two quotation symbols without anything in between, which means I am giving no instructions. And this has to be processed on the file called sample dot txt.

(Refer Slide Time: 11:22)




```
gphani@icme:~/scripts
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$ sed -e "" sample.txt
L1 linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice writer in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$ sed -e "" sample.txt
```


So, what would happen is the entire text file is reproduced on the screen, which means that the default action of stream editor is to read every line and do whatever processing that is asked, if there is nothing that is asked, then by default, it should just print it out. So, that is what has been done. You could start exploring the features of a stream editor.

(Refer Slide Time: 11:50)


```
gphani@icme:~/scripts$ sed -n -e "" sample.txt
gphani@icme:~/scripts$ sed -e "=" sample.txt
```



```
L10 array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$ sed -e "" sample.txt
gphani@icme:~/scripts$ sed -n -e "" sample.txt
gphani@icme:~/scripts$ sed -e "=" sample.txt
1
L1 linux is a free, opensource, secure and reliable os
2
L2 text editing: vim or emacs in place of notepad
3
L3 text processing: latex, libreoffice writer in place of microsoft word
4
L4 spreadsheet: libreoffice calc in place of microsoft excel
5
L5 slides: libreoffice impress in place of microsoft powerpoint
6
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
7
L7 browsing: mozilla firefox in place of microsoft edge
8
L8 email: mozilla thunderbird in place of microsoft outlook
9
L9 symbolic math: sympy, sagemath in place of maple, mathematica
10
```



```
L2 text editing: vim or emacs in place of notepad
3
L3 text processing: latex, libreoffice writer in place of microsoft word
4
L4 spreadsheet: libreoffice calc in place of microsoft excel
5
L5 slides: libreoffice impress in place of microsoft powerpoint
6
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
7
L7 browsing: mozilla firefox in place of microsoft edge
8
L8 email: mozilla thunderbird in place of microsoft outlook
9
L9 symbolic math: sympy, sagemath in place of maple, mathematica
10
L10 array math: octave, numpy, scilab in place of matlab
11
L11 literature: jabref, calibre in place of mendeley
12
L12 video editing: kdenlive in place of camtasia
13
L13 screen recording: obs in place of camtasia
14
L14 fetch files: wget, curl
15
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$
```



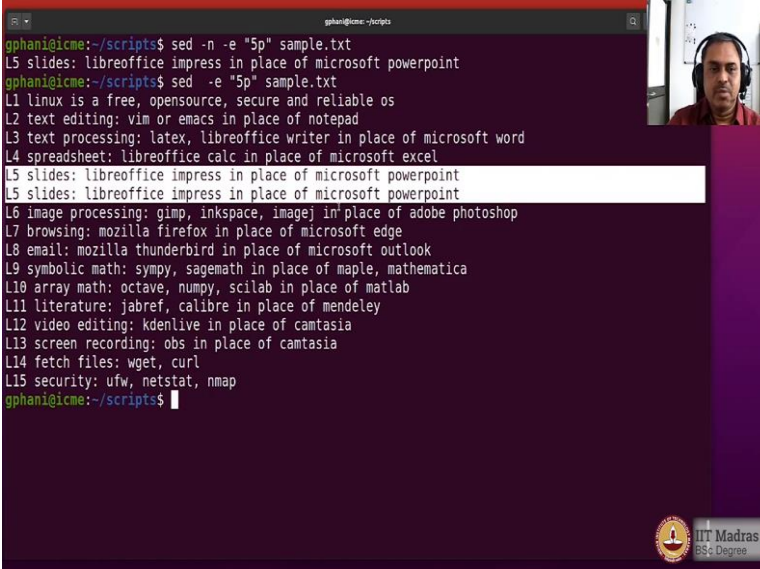
And here is one such feature where if you add minus n, it means that the default action of printing should not be performed. So, what do we expect when we run this command? You see that nothing happens, which means that there is no action that is specified within the quotes for the script, the default action of printing has been suppressed with the minus n option. So, this command does not give anything.

Now, these two extremes will tell us that stream editor can be told to print only those lines that match a particular pattern with which we will be doing some commands. And what to do with the remaining lines whether to print them or not. So, that way, we can decide what part of the input stream do we want to be sent on to the screen or to the pipe if he were to use sed in combination with other commands on the command line.

So, here is one usage where we would like to print the lines and equal to means that you are telling the stream editor to print the line number. And here you can see that there is a line number that is printed. And then there is a newline character. After that the first line has been printed, same way, the second line, and then the second line.

And then number 3, and then the third line. Number 5, and then the 5th line and so on. So, the equal to symbol prints the line number and then a newline character followed by the line that is read from the input string. This may be useful for some source code, if you want to send it to a printer, for example.

(Refer Slide Time: 13:28)

A terminal window with a dark background and light green text. The prompt is 'gphani@icme:~/scripts\$'. The command entered is 'sed -n -e "5p" sample.txt'. The output shows a list of software alternatives, with line 5 highlighted in white. The list includes: L5 slides: libreoffice impress in place of microsoft powerpoint, L1 linux is a free, opensource, secure and reliable os, L2 text editing: vim or emacs in place of notepad, L3 text processing: latex, libreoffice writer in place of microsoft word, L4 spreadsheet: libreoffice calc in place of microsoft excel, L5 slides: libreoffice impress in place of microsoft powerpoint, L6 image processing: gimp, inkspace, imagej in place of adobe photoshop, L7 browsing: mozilla firefox in place of microsoft edge, L8 email: mozilla thunderbird in place of microsoft outlook, L9 symbolic math: sympy, sagemath in place of maple, mathematica, L10 array math: octave, numpy, scilab in place of matlab, L11 literature: jabref, calibre in place of mendeley, L12 video editing: kdenlive in place of camtasia, L13 screen recording: obs in place of camtasia, L14 fetch files: wget, curl, L15 security: ufw, netstat, nmap. The prompt returns to 'gphani@icme:~/scripts\$'. In the top right corner, there is a small video feed of a man wearing headphones. In the bottom right corner, there is a logo for 'IIT Madras BS: Degree'.

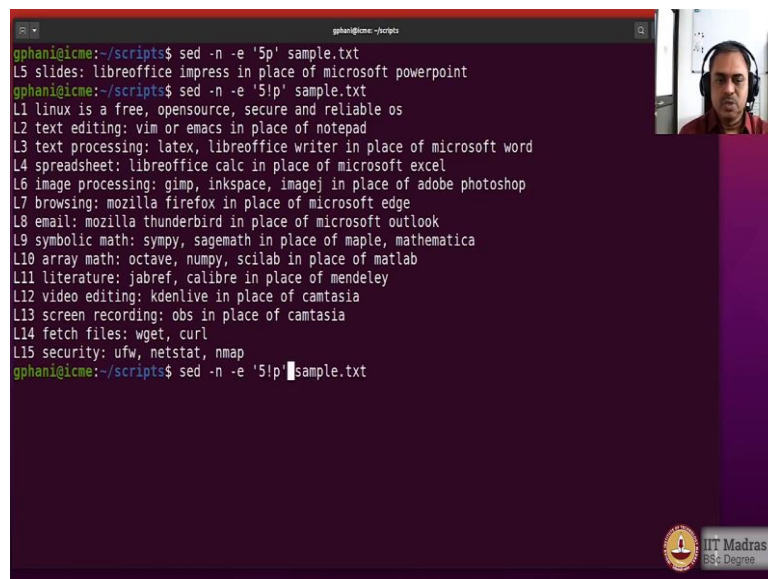
You can use the stream editor to pick a particular line or particular range of lines from the input stream. Here, we have an example of suppressing all other lines except the one which

has been specified in the command. And the command is 5p, the 5 refers to the address that means the line number on which the command has to be executed. And the action is p which means to print. So, what this line would say is to print the 5th line, and this says do not print anything else by default.

So, which means that the output of this is to print the 5th line, which you can see L5 and then the rest of it as per the output of the sample text file. Now, let us see if I skipped the minus n option what would happen, you will see that all other lines would also be printed, but the 5th line has been printed twice because the default action is to print and then the action I am specifically telling is also to print.

So, you have to use the minus n option whenever you are going to use filtering action so that you do not want the other lines which are not indicated to be matched by the address space that we are giving here. In which case it is the 5th line is the address space. So, if you do not want anything to be done on other lines, then you have to use minus n option.

(Refer Slide Time: 14:51)



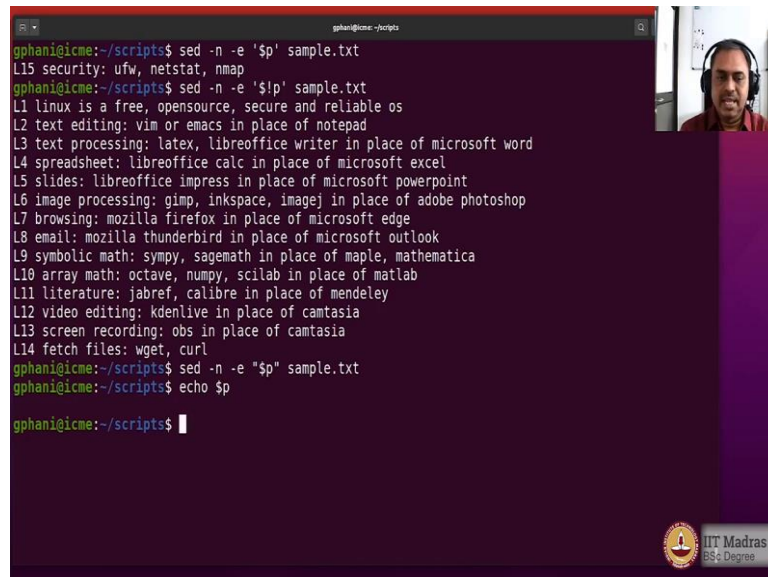
```
gphani@icme:~/scripts$ sed -n -e '5p' sample.txt
L5 slides: libreoffice impress in place of microsoft powerpoint
gphani@icme:~/scripts$ sed -n -e '5!p' sample.txt
L1 linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice writer in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$ sed -n -e '5!p' sample.txt
```

So, we repeat the previous command of printing the 5th line using single quotes. Single quotes would do the same action as double quotes at except that there are some special characters, such as the exclamation mark or the dollar symbol, which should be interpreted by the shell if you want to escape the interpretation by the shell and pass on characters directly to the sed command then single quotes are useful.

So, here for example, if I put an exclamation mark, it means that I am going to print only those lines which do not match the address 5, which means every line except the 5th line.

And you can see here that after the fourth line, it is the 6th line that has been printed. So, you can skip some lines by using the address and then an exclamation mark to negate that particular address. Some more such examples.

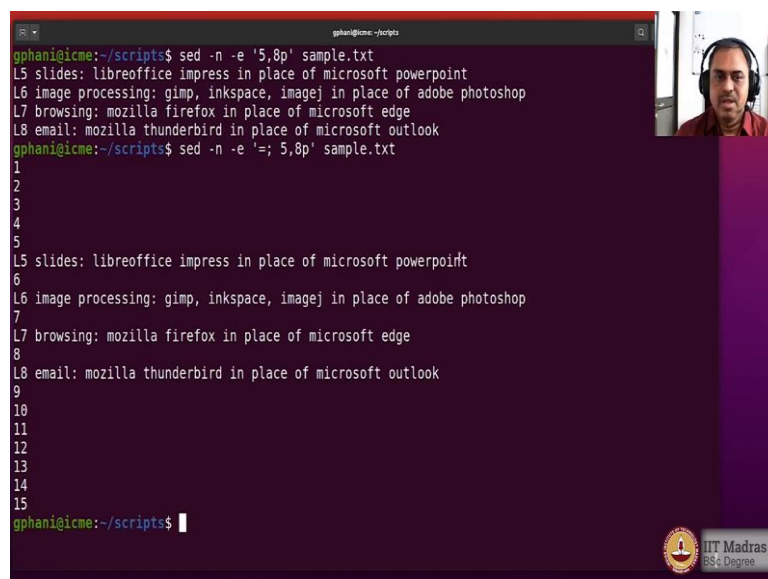
(Refer Slide Time: 15:40)

A terminal window with a dark purple background. The prompt is 'gphani@icme: ~/scripts'. The user enters 'sed -n -e '\$p' sample.txt'. The output shows lines 15 through 14 of 'sample.txt'. Line 15 is 'security: ufw, netstat, nmap'. Line 16 is 'linux is a free, open source, secure and reliable os'. Line 17 is 'text editing: vim or emacs in place of notepad'. Line 18 is 'text processing: latex, libreoffice writer in place of microsoft word'. Line 19 is 'spreadsheet: libreoffice calc in place of microsoft excel'. Line 20 is 'slides: libreoffice impress in place of microsoft powerpoint'. Line 21 is 'image processing: gimp, inkspace, imagej in place of adobe photoshop'. Line 22 is 'browsing: mozilla firefox in place of microsoft edge'. Line 23 is 'email: mozilla thunderbird in place of microsoft outlook'. Line 24 is 'symbolic math: sympy, sagemath in place of maple, mathematica'. Line 25 is 'array math: octave, numpy, scilab in place of matlab'. Line 26 is 'literature: jabref, calibre in place of mendeley'. Line 27 is 'video editing: kdenlive in place of camtasia'. Line 28 is 'screen recording: obs in place of camtasia'. Line 29 is 'fetch files: wget, curl'. The user then enters 'sed -n -e '\$p' sample.txt' and 'echo \$p'. The output is 'gphani@icme:~/scripts\$'. In the bottom right corner, there is a logo for IIT Madras BS: Degree.

```
gphani@icme:~/scripts$ sed -n -e '$p' sample.txt
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$ sed -n -e '$!p' sample.txt
L1 linux is a free, open source, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice writer in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
gphani@icme:~/scripts$ sed -n -e '$p' sample.txt
gphani@icme:~/scripts$ echo $p
gphani@icme:~/scripts$
```

So, dollar would mean the last line. And except the last line, then Dollar Bank would give you that. Now, what happens if I happen to use single quotes? You can see that it would then be interpreted by the shell and the variable p will be substituted there and then there is a problem because dollar p, there is no variable by name p. So, you can see dollar p is nothing. And therefore, we have passed on a null to the script and therefore nothing has come onto the screen.

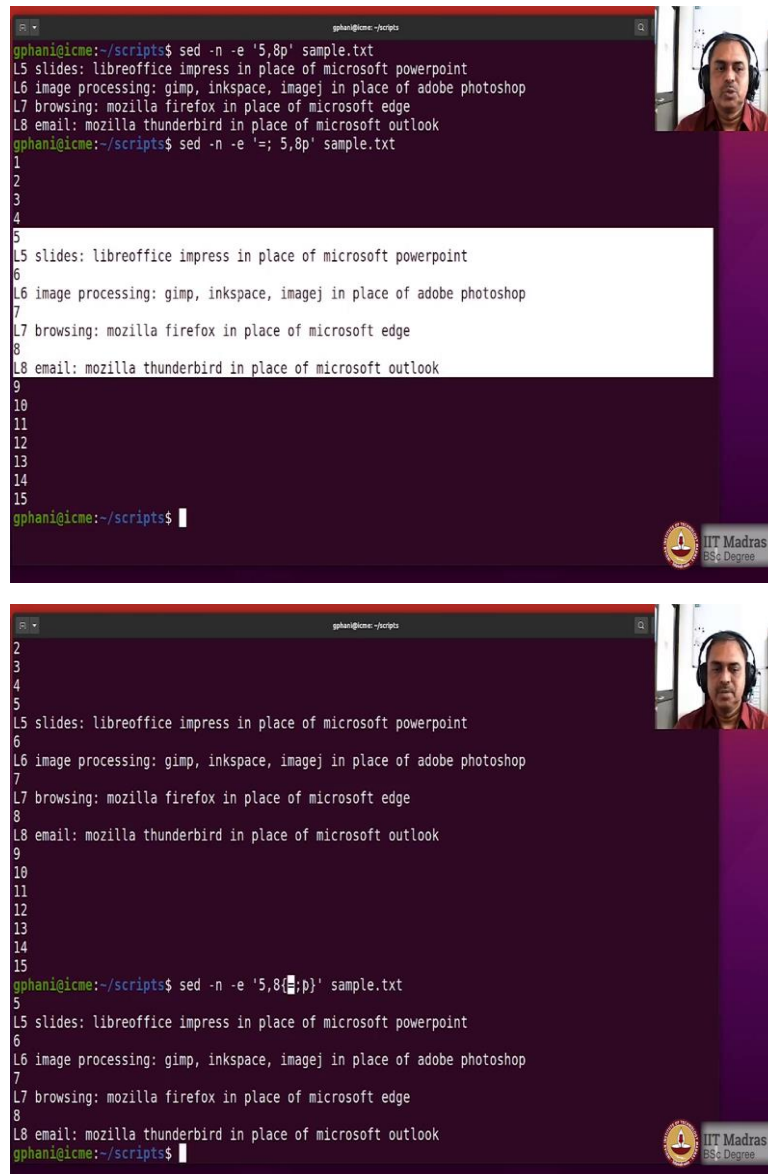
(Refer Slide Time: 16:17)

A terminal window with a dark purple background. The prompt is 'gphani@icme: ~/scripts'. The user enters 'sed -n -e '\$!p' sample.txt'. The output shows lines 15 through 14 of 'sample.txt'. Line 15 is 'slides: libreoffice impress in place of microsoft powerpoint'. Line 16 is 'image processing: gimp, inkspace, imagej in place of adobe photoshop'. Line 17 is 'browsing: mozilla firefox in place of microsoft edge'. Line 18 is 'email: mozilla thunderbird in place of microsoft outlook'. The user then enters 'sed -n -e '\$!p' sample.txt'. The output shows lines 1 through 15 of 'sample.txt'. Line 1 is '1'. Line 2 is '2'. Line 3 is '3'. Line 4 is '4'. Line 5 is '5'. Line 6 is '6'. Line 7 is '7'. Line 8 is '8'. Line 9 is '9'. Line 10 is '10'. Line 11 is '11'. Line 12 is '12'. Line 13 is '13'. Line 14 is '14'. Line 15 is '15'. The user then enters 'sed -n -e '\$!p' sample.txt'. The output is 'gphani@icme:~/scripts\$'. In the bottom right corner, there is a logo for IIT Madras BS: Degree.

```
gphani@icme:~/scripts$ sed -n -e '$!p' sample.txt
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
gphani@icme:~/scripts$ sed -n -e '$!p' sample.txt
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
gphani@icme:~/scripts$
```


Let us see how to give an address range for action. So, let us say 5 to 8 p. So, I am asking the lines from the 5th one to the 8th one including both of these two to be printed. And if I want to combine another command here, I could do that as follows. So, equal to is one action, which means print the line number. And from the address space 5 to 8, print the line also. So, it means that for the 1st to 4th and then 9th onwards only the line number has been printed, but from the 5th to 8th the line itself has been printed.

(Refer Slide Time: 17:05)



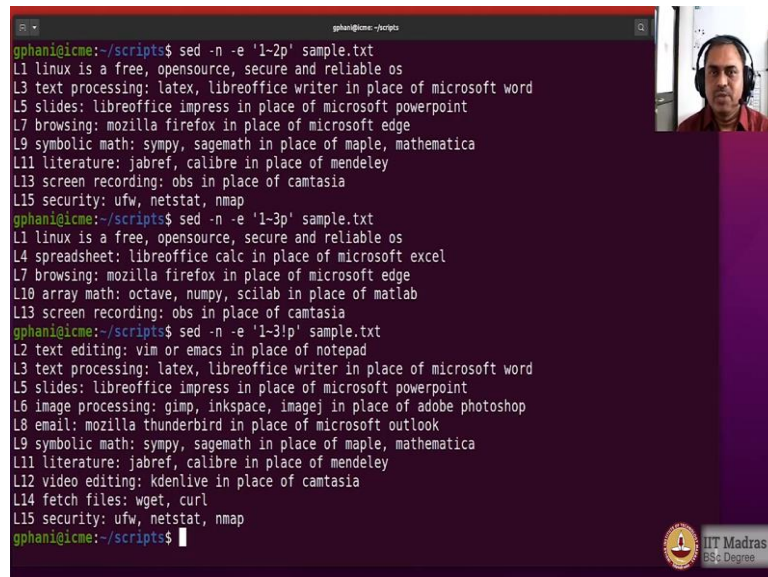
```
gphani@icme:~/scripts$ sed -n -e '5,8p' sample.txt
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
gphani@icme:~/scripts$ sed -n -e '5,8p' sample.txt
1
2
3
4
5
L5 slides: libreoffice impress in place of microsoft powerpoint
6
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
7
L7 browsing: mozilla firefox in place of microsoft edge
8
L8 email: mozilla thunderbird in place of microsoft outlook
9
10
11
12
13
14
15
gphani@icme:~/scripts$
```

```
gphani@icme:~/scripts$ sed -n -e '5,8{[p]}' sample.txt
5
L5 slides: libreoffice impress in place of microsoft powerpoint
6
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
7
L7 browsing: mozilla firefox in place of microsoft edge
8
L8 email: mozilla thunderbird in place of microsoft outlook
gphani@icme:~/scripts$
```

Now, sometimes we may want only this part of the text to come, so that the remaining lines where the line is not being printed, the line number also should not come. So, that could be achieved in this manner. So, the address range is given for which a set of actions can be specified. And you see that this would give you what you wanted, namely, for the address

range 5 to 8 two actions are specified, one is to print the line number, the second is to print the line also.

(Refer Slide Time: 17:39)

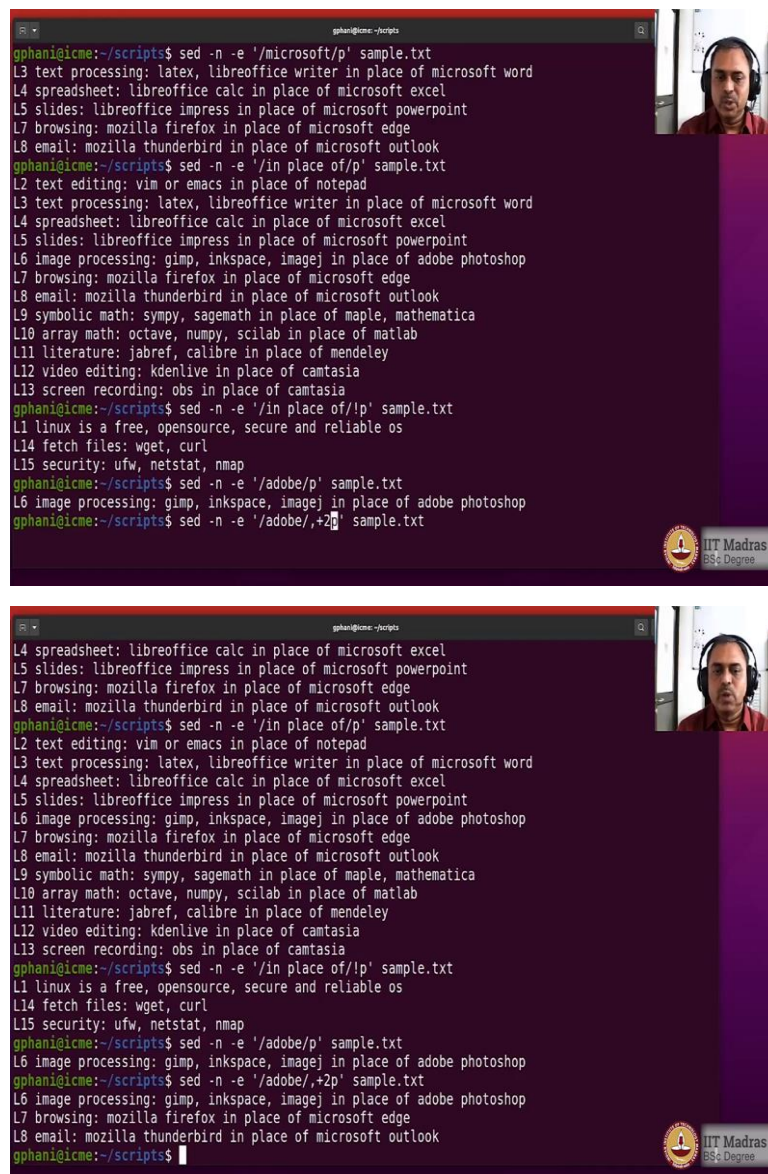


```
gphani@icme:~/scripts$ sed -n -e '1-2p' sample.txt
L1 linux is a free, opensource, secure and reliable os
L2 text processing: latex, libreoffice writer in place of microsoft word
gphani@icme:~/scripts$ sed -n -e '1-3p' sample.txt
L1 linux is a free, opensource, secure and reliable os
L2 text processing: latex, libreoffice writer in place of microsoft word
L3 text editing: vim or emacs in place of notepad
gphani@icme:~/scripts$ sed -n -e '1-3!p' sample.txt
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$
```

Now, you could do things that are not very easy in head and tail commands, namely to hop every nth line and print the remaining ones. So, I would go from the first line, and then the till that tells sed that the number which follows it should be the step. So, every second line, I would like to print from the file sample dot txt. So, you could see that every odd line has been printed. You could change the hopping from 2 to 3. So, every 4th line has been printed.

You could also give a negation here and you will see that the remaining lines would then be printed, so you could choose to print or not every nth as a hopping. Sometimes this is useful in skipping some of the lines from an input stream which contains numbers. So, you have a spreadsheet of let us say a million numbers and you want to sample every second or every third line for further processing. So, you could use this kind of a command for such filtering.

(Refer Slide Time: 18:52)



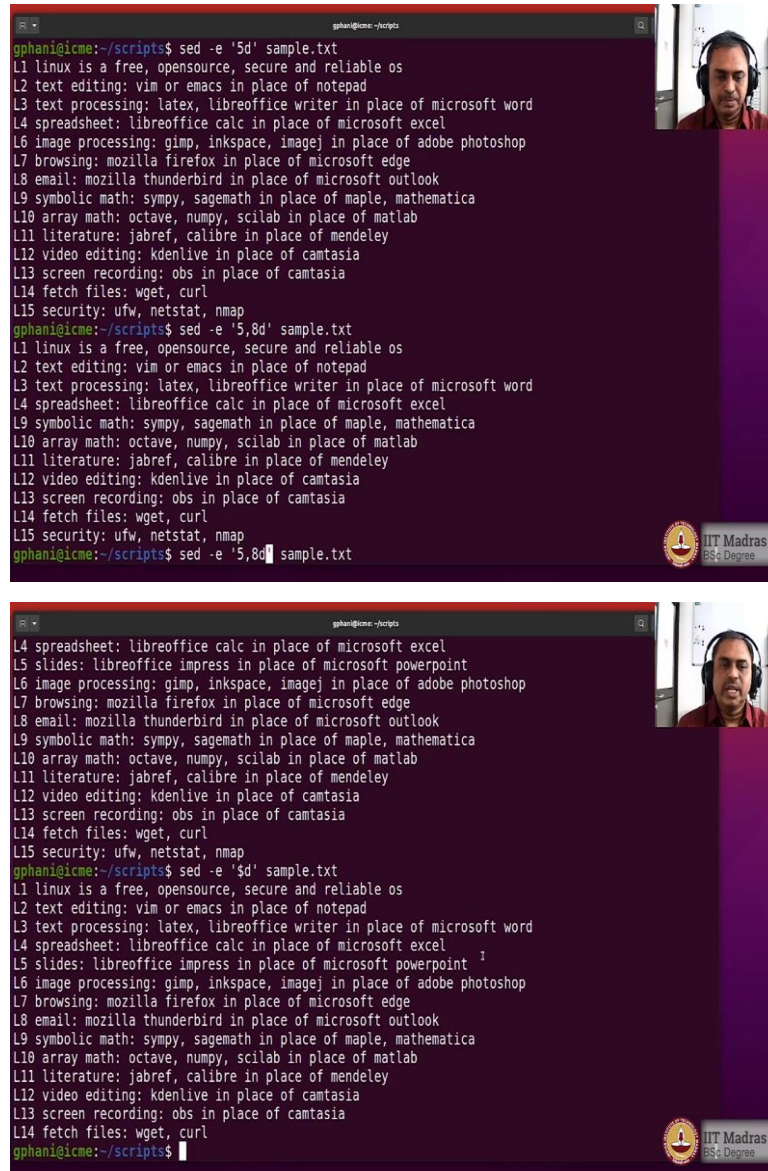
```
gphani@icme:~/scripts$ sed -n -e '/microsoft/p' sample.txt
L3 text processing: latex, libreoffice writer in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
gphani@icme:~/scripts$ sed -n -e '/in place of/p' sample.txt
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice writer in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
gphani@icme:~/scripts$ sed -n -e '/in place of/p' sample.txt
L1 linux is a free, opensource, secure and reliable os
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$ sed -n -e '/adobe/p' sample.txt
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
gphani@icme:~/scripts$ sed -n -e '/adobe/,+2p' sample.txt
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
gphani@icme:~/scripts$
```

The address can also be given using a regular expression. So, here, what I do is I give a phrase and an action. So, the action is to print the phrase is Microsoft. So, every line that contains that particular word will be printed. The regular expression is honoured with the spaces also in between. So, you could also say that every line which contains a phrase in place of should be printed.

And you could also ask the negation of it using the exclamation mark. So, those links which do not contain the phrase in place of will be printed here. Then you could also do an action where you could specify how many lines you want to print after the matching line has been shown. So, here I am showing the line that is matching the phrase adobe. And let us say I want to give a range and say that two lines after the line matching adobe have to be printed.

So, you can see that the 6th line matches adobe and then after the two more lines have been printed.

(Refer Slide Time: 20:01)



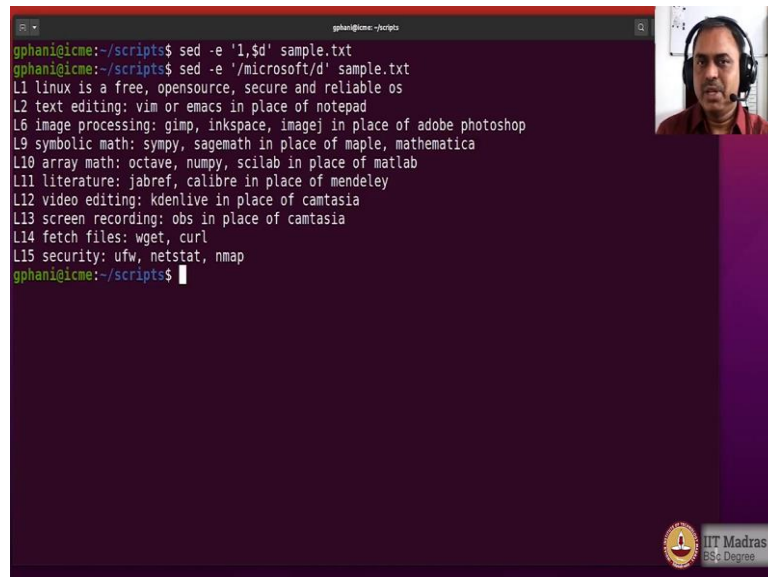
```
gphani@icme:~/scripts$ sed -e '5d' sample.txt
L1 linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice writer in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$ sed -e '5,8d' sample.txt
L1 linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice writer in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$ sed -e '5,8d' sample.txt
```

Now, for the purpose of filtering lines, one of the actions that you could do is to actually delete. So, you can delete a particular line and then send it onto the screen or the standard out for further processing by other commands. So, let us try that out. So, I would say 5d. So, what would happen is that the 5th line has been deleted, you can see that the L4 after the L6 is coming up.

So, 5th line has been deleted and then the default action of printing has been maintained, because we are not specifying the minus n option. You could specify a range also. So, from the 5th to 8th lines is deleted here. So, you could see that after the fourth it is the 9th line

that is coming up. You could also say I do not want the first line or you could say I do not want the last line. So, the last line is the 15th one, so up to 14th we are printing. And in the case of deleting the first 10 the second onwards will be printing.

(Refer Slide Time: 21:19)

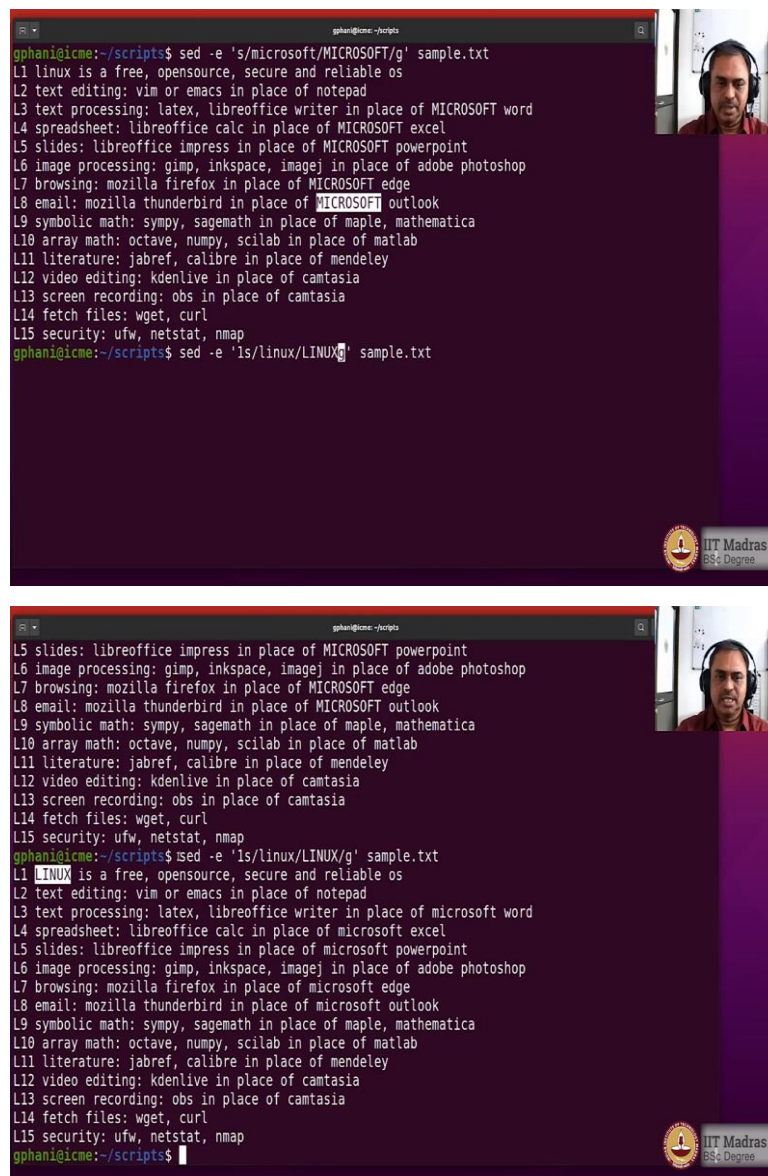


```
gphani@icme:~/scripts$ sed -e '1,$d' sample.txt
gphani@icme:~/scripts$ sed -e '/microsoft/d' sample.txt
L1 linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$
```

What happens when we give a range in this manner, from 1 to dollar d. So, you will see that the from the first to the last line deleted, so naturally nothing will be shown on the screen. You could also say that using a regular expression. So, all those lines which mentioned microsoft should be deleted. So, you could do that and you will see the remaining lines are shown on the screen.

So, apart from printing a particular set of lines by either numbering the address space or by using pattern matching or deleting some of the lines from printing, you could also do one action that is very popular using the stream editor namely to substitute a particular phrase with another phrase. So, let us explore that because it is the most popular usage of sed command.

(Refer Slide Time: 22:12)



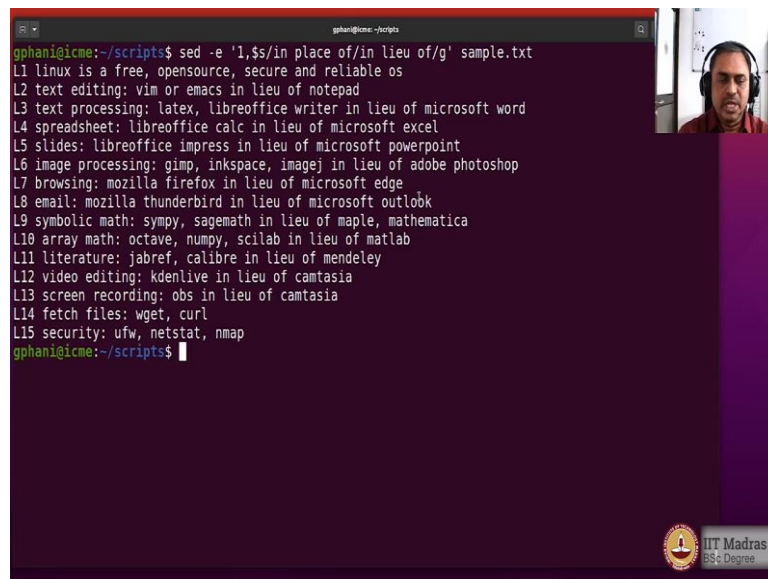
```
gphani@icme:~/scripts$ sed -e 's/microsoft/MICROSOFT/g' sample.txt
L1 linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice writer in place of MICROSOFT word
L4 spreadsheet: libreoffice calc in place of MICROSOFT excel
L5 slides: libreoffice impress in place of MICROSOFT powerpoint
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of MICROSOFT edge
L8 email: mozilla thunderbird in place of MICROSOFT outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$ sed -e 's/linux/LINUX/g' sample.txt
L5 slides: libreoffice impress in place of MICROSOFT powerpoint
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of MICROSOFT edge
L8 email: mozilla thunderbird in place of MICROSOFT outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$ sed -e 's/linux/LINUX/g' sample.txt
L1 LINUX is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice writer in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$
```

So, consider this command where we are asking without an address space here. So, if you see `s` is a command, so before the command, you are supposed to see an address space, and I am not specifying which means that it is applicable for every line. And the thing that is coming after that is all, whatever is coming after `s` is all options for the particular command. So, `s` is a search and replace.

So, the options are to, the first part is a regular expression to find out what you are going to search and match. And the second part is to replace and the `g` means that you must replace it for any number of occurrences on the line. So, wherever the small case `microsoft` is appearing, it will be replaced with capital letters and you could see that it has happened here on-line number 3 onwards up to line number 8.

You could also ask this kind of a replacement on a specific line. So, on the first line you have Linux, so you would like that let us say to be replaced. So, on the first line substitute the small case Linux with capital case Linux and you will see that the entire extreme is being sent except that there is a change that has happened on the first line the word Linux has become in capital letters. This is one way by which sed editor is used in a common manner because you are actually processing a specific transformation of the text before it is sent out of the screen.

(Refer Slide Time: 23:48)

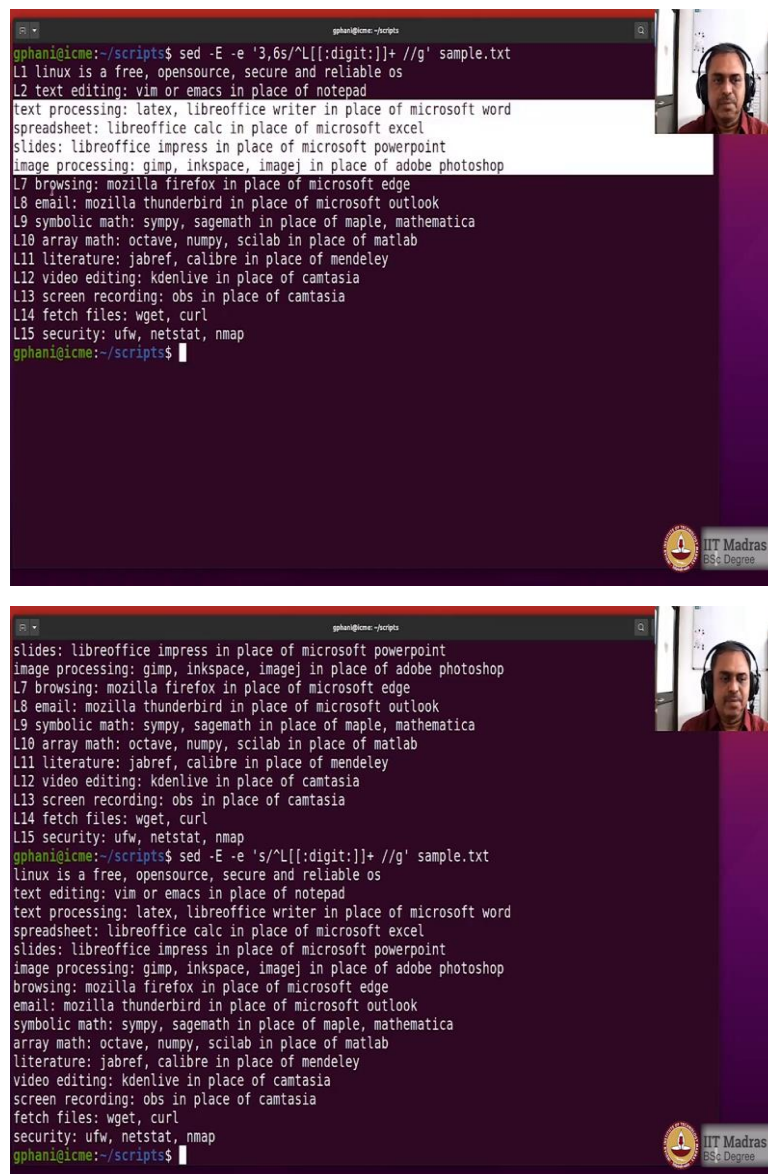


```
gphani@icme:~/scripts$ sed -e '1,$s/in place of/in lieu of/g' sample.txt
L1 linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in lieu of notepad
L3 text processing: latex, libreoffice writer in lieu of microsoft word
L4 spreadsheet: libreoffice calc in lieu of microsoft excel
L5 slides: libreoffice impress in lieu of microsoft powerpoint
L6 image processing: gimp, inkspace, imagej in lieu of adobe photoshop
L7 browsing: mozilla firefox in lieu of microsoft edge
L8 email: mozilla thunderbird in lieu of microsoft outlook
L9 symbolic math: sympy, sagemath in lieu of maple, mathematica
L10 array math: octave, numpy, scilab in lieu of matlab
L11 literature: jabref, calibre in lieu of mendeley
L12 video editing: kdenlive in lieu of camtasia
L13 screen recording: obs in lieu of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$
```

Let us say I do not want to use the phrase in place of and I want to use a phrase like in lieu of. And we would like it to be in an address space from first line to the last line. So, you will see that all the 15 lines have been printed, but the phrase in place off is now replaced as in lieu of. So, you can see that some of the parts of the line can be modified and all the lines can then be also passed our to the standard output by sed command.

And what we are doing here is such and replace one phrase with another phrase and this is a very common usage of sed editor. Now, let us say we would like to print these lines without the L followed by the digit after that in the beginning of the line, so I want to remove the first couple of characters. So, this can also be achieved by using the regular expression engine, the extended set will be used because the character set for alphanumeric matching can be used for the beginning of the text. So, let us try that out.

(Refer Slide Time: 25:02)



```
gphani@icme: ~/scripts
gphani@icme:~/scripts$ sed -E -e '3,6s/^L[[:digit:]]+ //' sample.txt
L1 linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
text processing: latex, libreoffice writer in place of microsoft word
spreadsheet: libreoffice calc in place of microsoft excel
slides: libreoffice impress in place of microsoft powerpoint
image processing: gimp, inkspace, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$
```

```
gphani@icme: ~/scripts
gphani@icme:~/scripts$ sed -E -e '3,6s/^L[[:digit:]]+ //' sample.txt
linux is a free, opensource, secure and reliable os
text editing: vim or emacs in place of notepad
text processing: latex, libreoffice writer in place of microsoft word
spreadsheet: libreoffice calc in place of microsoft excel
slides: libreoffice impress in place of microsoft powerpoint
image processing: gimp, inkspace, imagej in place of adobe photoshop
browsing: mozilla firefox in place of microsoft edge
email: mozilla thunderbird in place of microsoft outlook
symbolic math: sympy, sagemath in place of maple, mathematica
array math: octave, numpy, scilab in place of matlab
literature: jabref, calibre in place of mendeley
video editing: kdenlive in place of camtasia
screen recording: obs in place of camtasia
fetch files: wget, curl
security: ufw, netstat, nmap
gphani@icme:~/scripts$
```

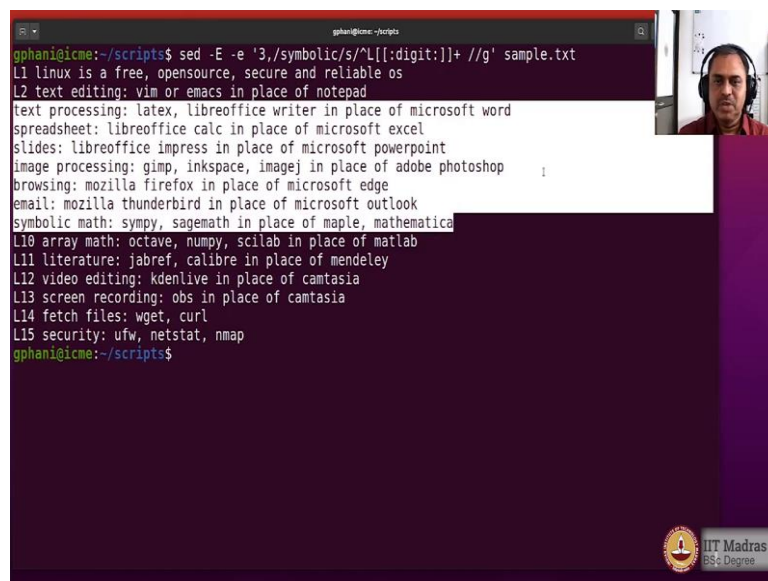
So, minus capital E indicates that you are going to use the Extended Regular Expression Engine so that the character set with the colons on either ends could be used. And what I am trying to do here is for an address range letter, say from 3 to 6 lines, I am going to substitute in the beginning of the line, a capital L followed by any digit multiple times and then a space, and I am going to remove it, so I am giving null there and this is occurring on the file called sample dot txt.

So, from the 3rd line to the 6th line, perform a command which is search and replace. And what to search for, at the beginning of the line a capital L should be followed by a digit which can occur multiple times followed by a space and that has to be replaced with the

null. And you can then see what does it do, you can see that the line number 3 to 6 have been trimmed in the beginning, where L3 up to L6 have been cut off.

So, this is an illustration of modifying the incoming stream using the Extended Regular Expression Engine. If you want this to be applicable for all the lines, you simply remove the address range. So, by default it will be for every line. And you see that for all the lines the initial part of the line with capital L followed by the line number have been removed.

(Refer Slide Time: 26:52)



```
gphani@icme:~/scripts$ sed -E -e '3,/symbolic/s/^L[[[:digit:]]+ //' sample.txt
L1 linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
text processing: latex, libreoffice writer in place of microsoft word
spreadsheet: libreoffice calc in place of microsoft excel
slides: libreoffice impress in place of microsoft powerpoint
image processing: gimp, inkspace, imagej in place of adobe photoshop
browsing: mozilla firefox in place of microsoft edge
email: mozilla thunderbird in place of microsoft outlook
symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$
```

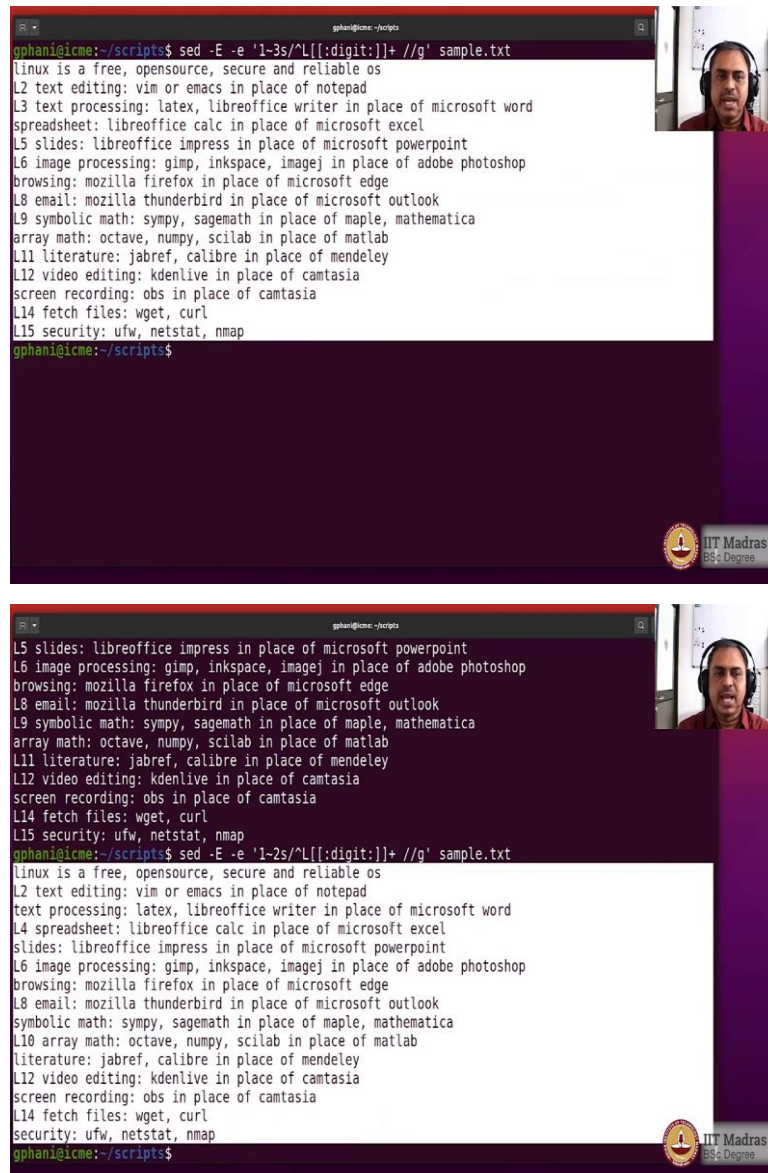
While in the POSIX version of sed the address range is restricted in the GNU extended feature set, the address range can be such that the second part can also be the regular expression. So, let us try that out. So, what are we doing here? So, here we are saying that the er e engine has to be used and the script is coming on the command line and this has to be affected from the 3rd line until a line in which the phrase symbolic is occurring.

And the action that has to be taken is to substitute. And what has to be substituted? At the beaming of the line a capital L has to come after that a digit can come and let us say I would also want multiple such digits to come and the space and that has to be removed. So, you could see that what is happening is from the 3, line 3 until the last line here 9th line which contains a word symbolic, what we are doing is removing the initial portion of the line which contains capital L followed by number.

So, you could see that the address range is now specified where the first part is a number 3rd line specifically, the 2nd part is a pattern which says whichever line after the 3rd one,

which contains the phrase symbolic can be used. Now, you could also combine this feature of stepping with the extended regular expression matching and replacing.

(Refer Slide Time: 28:22)



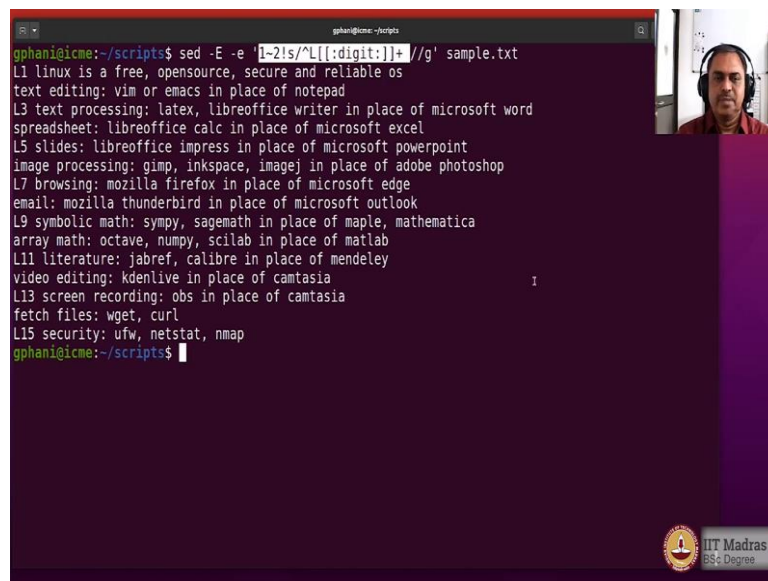
The image shows two screenshots of a terminal window. The top screenshot shows the command `sed -E -e '1-3s/^L[[:digit:]]+ //' sample.txt` being executed. The output shows a list of software alternatives, with the first three characters of every third line (lines 3, 6, 9, 12, 15) being removed. The bottom screenshot shows the same command being executed again, but with a different sed script: `sed -E -e '1-2s/^L[[:digit:]]+ //' sample.txt`. This time, the first two characters of every third line are removed. Both screenshots include a video feed of a person in the top right corner and an IIT Madras logo in the bottom right corner.

```
gphani@icme:~/scripts$ sed -E -e '1-3s/^L[[:digit:]]+ //' sample.txt
linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice writer in place of microsoft word
spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$

gphani@icme:~/scripts$ sed -E -e '1-2s/^L[[:digit:]]+ //' sample.txt
linux is a free, opensource, secure and reliable os
text processing: latex, libreoffice writer in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy, scilab in place of matlab
literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
screen recording: obs in place of camtasia
L14 fetch files: wget, curl
security: ufw, netstat, nmap
gphani@icme:~/scripts$
```

So, what we do here is from the first line, every third line we would perform the substitution whatever we mentioned just now. So, you see that the first line onwards, every 4th line because three are being skipped. So, stepping every 4th line, we are seeing that the initial part of the line has been trimmed away. You could do that with every second line. So, you will see that every odd number line has been trimmed of the initial portion of the text, how it is mentioned here.

(Refer Slide Time: 29:14)

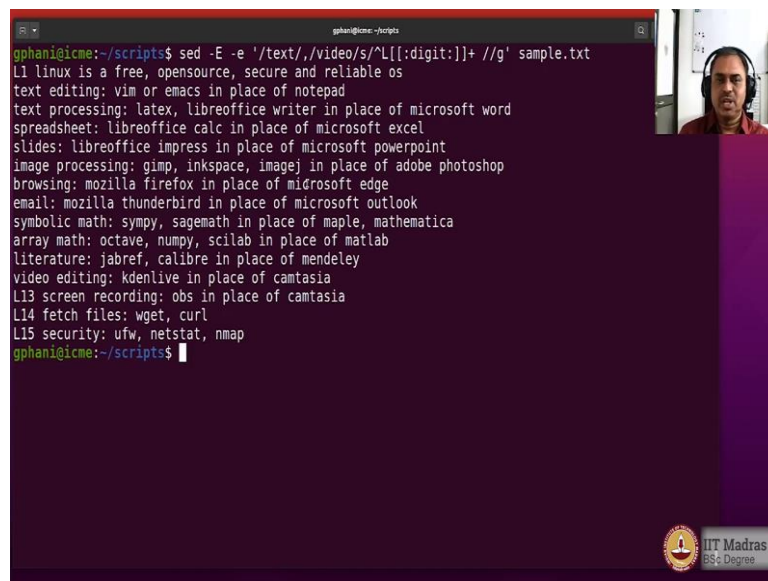


```
gphani@icme:~/scripts$ sed -E -e 's/^L[0-9]* //' sample.txt
L1 linux is a free, opensource, secure and reliable os
text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice writer in place of microsoft word
spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
image processing: gimp, inkspace, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$
```

And you could combine this with the negation of the address range also where the opposite is done, where the odd number lines are left as it is, but the even number lines have been trimmed in the beginning to remove the capital L followed by a number. So, you could see that a fairly complicated action is performed here where from the first line onwards every second line we would like to perform some action and that action is to substitute the initial portion of L followed by digit, it is nothing.

And the bank says that such lines that is from the first every second have to be skipped. So, from the second onwards, every second line will be acted upon. And that is what we are seeing here as every even line number is actually processed to remove the initial couple of characters of the line.

(Refer Slide Time: 30:09)



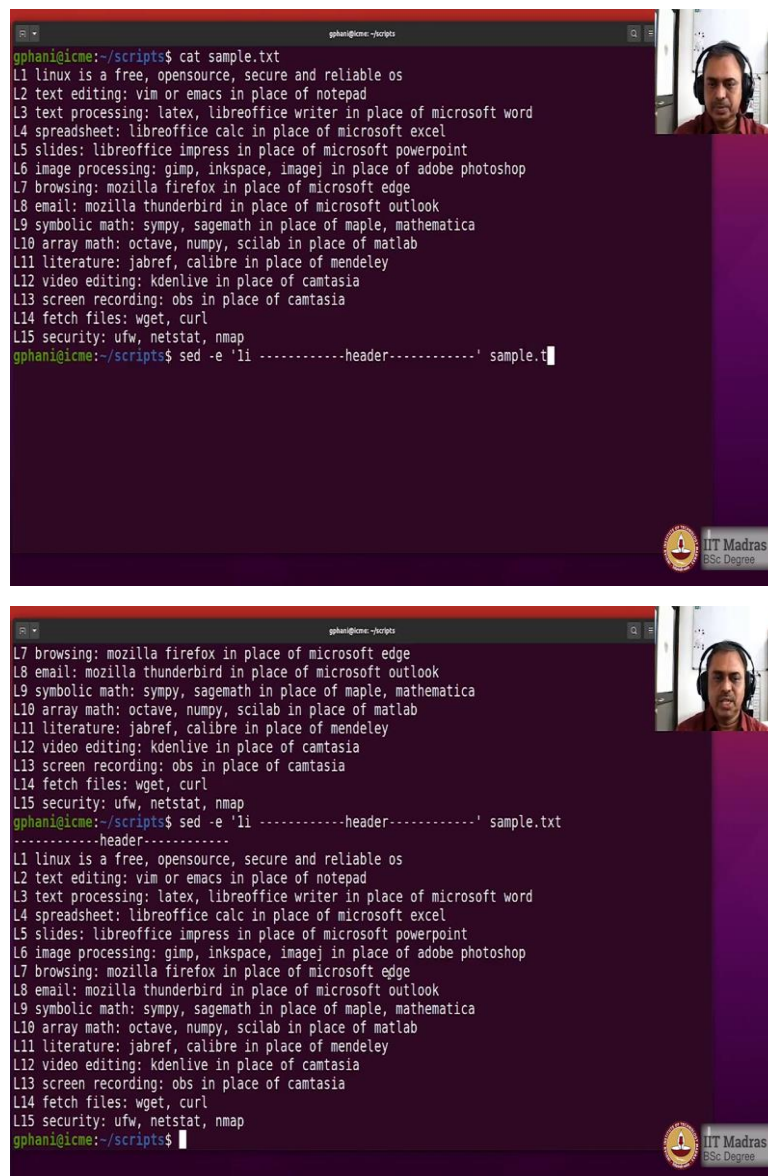
```
gphani@icme:~/scripts$ sed -E -e '/text/,/video/s/^L[[[:digit:]]+ //'g' sample.txt
L1 linux is a free, opensource, secure and reliable os
text editing: vim or emacs in place of notepad
text processing: latex, libreoffice writer in place of microsoft word
spreadsheet: libreoffice calc in place of microsoft excel
slides: libreoffice impress in place of microsoft powerpoint
image processing: gimp, inkspace, imagej in place of adobe photoshop
browsing: mozilla firefox in place of microsoft edge
email: mozilla thunderbird in place of microsoft outlook
symbolic math: sympy, sagemath in place of maple, mathematica
array math: octave, numpy, scilab in place of matlab
literature: jabref, calibre in place of mendeley
video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$
```

In the GNU extended feature set the address range can be such that both the parts of the starting and the ending can also be regular expressions. So, let us try that out here. So, the starting part I would say is a phrase text should appear, the ending part should be a line that has video and what should I do in that I should do the action which I have mentioned namely to remove the text in between, so capital L.

So, from the line that contains a word text to the line that contains the word video, remove the initial portion of the line. So, you could see that the first line on which text word appears is the second one. And the last line where the video word is appearing is the 12th one. And for those lines, the action that was performed was to substitute the initial couple of characters to remove L followed by the digit and the space that follows the number.

So, you can see that this is a fairly intricate action, where some lines are being skipped, some lines are being processed and again some other lines that skipped. So, if you have a text file in which some comments are there, and then some specific fields are there which have to be modified, you could actually use this kind of a address range to ensure that you do not have to worry about the line numbers, but only by the occurrence of those particular text strings in the file to decide from which line to which line you would like to act on.

(Refer Slide Time: 31:55)



The image consists of two screenshots of a terminal window, likely from a video lecture. The terminal shows a user named 'gphani' at a machine named 'icme' in the directory '/scripts'. The user has created a file named 'sample.txt' containing 15 lines of text, each starting with a line number (L1 to L15) followed by a description of a software alternative. In the first screenshot, the user runs the command 'sed -e '1i -----header-----' sample.txt'. The second screenshot shows the result: the file 'sample.txt' now has a header line inserted at the beginning, before the first line of the original text. The IIT Madras logo is visible in the bottom right corner of the terminal window.

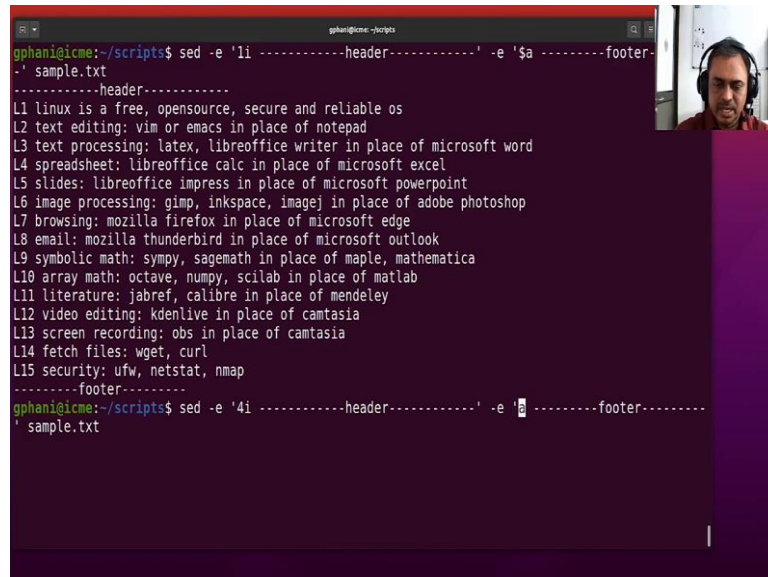
```
gphani@icme:~/scripts$ cat sample.txt
L1 linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice writer in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$ sed -e '1i -----header-----' sample.txt
-----header-----
L1 linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice writer in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$
```

Now, there are some more actions that are possible apart from printing, deleting and substitution you also have the possibilities to append or insert text. So, let us take the example where we would like to insert a header and a footer to the lines of text. So, what are the lines of the text? So, you could see that there are 15 lines. So, at the first line at the last line, we would like to insert some header and footer.

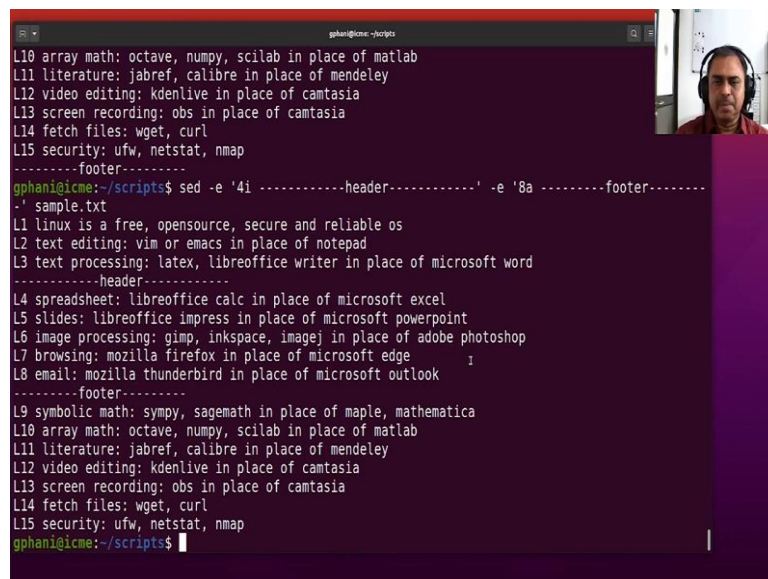
This can be achieved using the echo command and cat command in combination. However, we would like to do it with a sed so that we can understand where actually the insertion of text can take place. So, the address where insertion has to happen is on line number 1 and li means that start inserting.

And what do we want to insert? Let us say if phrase which looks like a header there and I close the court. And then so, what is it do, it prints the header inserts it at the first line. And I can actually use multiple scripts on the same command line I can print the header as well as footer one by one.

(Refer Slide Time: 33:07)



```
gphani@icme:~/scripts$ sed -e '1i -----header-----' -e '$a -----footer-'
-' sample.txt
-----header-----
L1 linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice writer in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
-----footer-----
gphani@icme:~/scripts$ sed -e '4i -----header-----' -e '$a -----footer-'
-' sample.txt
```

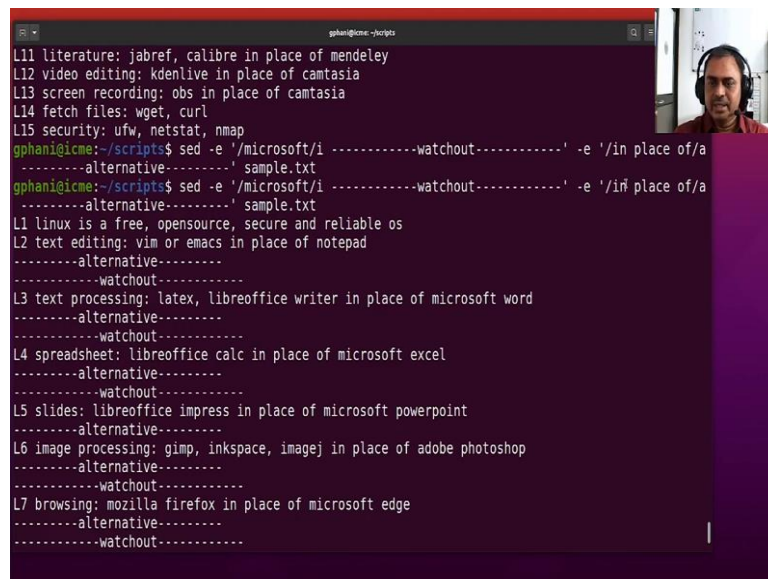


```
L10 array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
-----footer-----
gphani@icme:~/scripts$ sed -e '4i -----header-----' -e '8a -----footer-----'
-' sample.txt
-----header-----
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
-----footer-----
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$
```

So, I would have minus e once more coming up and the dollar says the last line and again append. So, append means after the last line. And what do I want to append? I can append say this is footer. So, now you see that this command shows you how to add a line in front of the first one and after the last one. So, i means insert before the first line, a means append after the last line.

And whatever you want to insert before the first line is given here, and whatever you want to append after the last line is given here. Now, you could also do this at any line number, which is also something that is quite powerful, 4th line, let us say 8th line. So, you could see that I have split the now text into three parts, before the fourth line I have put up a header line and after the eighth line I have put up a footer line. Naturally, we can do these actions of insertion and appending using any other address range patterns that we have seen earlier.

(Refer Slide Time: 34:25)



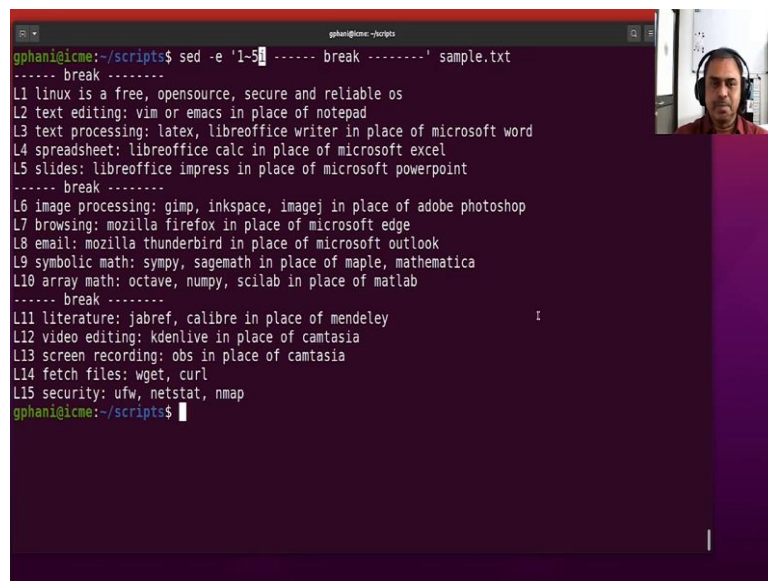
```
gphani@icme:~/scripts$ sed -e '/microsoft/i -----watchout-----' -e '/in place of/a -----alternative-----' sample.txt
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
-----watchout-----
L1 linux is a free, opensource, secure and reliable os
-----alternative-----
L2 text editing: vim or emacs in place of notepad
-----alternative-----
-----watchout-----
L3 text processing: latex, libreoffice writer in place of microsoft word
-----alternative-----
-----watchout-----
L4 spreadsheet: libreoffice calc in place of microsoft excel
-----alternative-----
-----watchout-----
L5 slides: libreoffice impress in place of microsoft powerpoint
-----alternative-----
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
-----alternative-----
-----watchout-----
L7 browsing: mozilla firefox in place of microsoft edge
-----alternative-----
-----watchout-----
```

So, let us say for every line that contains the phrase microsoft, I would like to insert a warning saying that watch out. Every line that contains in place of, I would like to append a line say alternative. So, look at it what we are doing for every line that contains the phrase microsoft I am inserting a line before it which says watchout.

And for every line that contains in place of I am actually attending a line alternative. So, you will see that many, many such lands will be inserted because these two phrases are occurring many times and you could see here that in place of has come here and therefore, immediately there is an attending of the line alternative coming up to that.

And here is a microsoft that is appearing. So, intent of it you have got watchout. And in place of it has come here, so therefore, alternative has come after that. So, you can actually combine the insertion or appending with the address space possibilities that we have to achieve any kind of breakage of the input stream as you like.

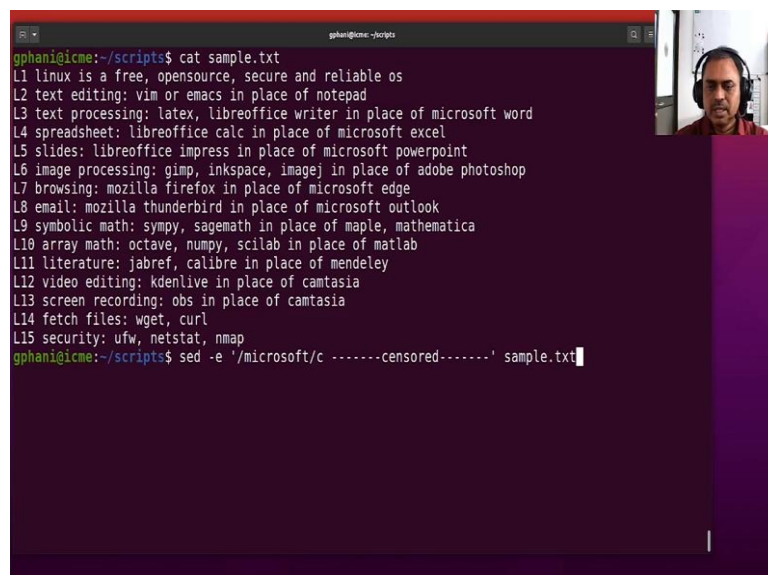
(Refer Slide Time: 35:44)



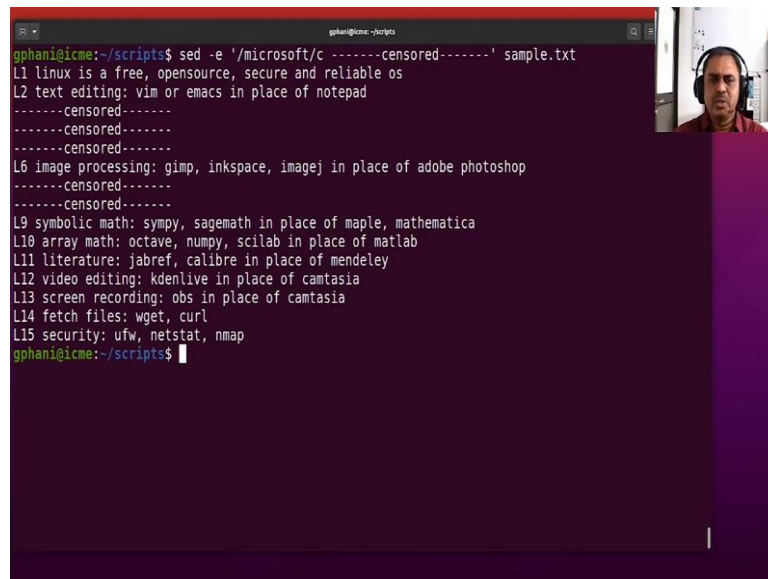
```
gphani@icme:~/scripts$ sed -e '1-5!----- break -----' sample.txt
----- break -----
L1 linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice writer in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
----- break -----
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy, scilab in place of matlab
----- break -----
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$
```

For example, I would like to print a break after every 5th line using sed we could do that here quite nicely. So, you can see that after every 5 lines, there is a break that we have inserted because the tilde operator tells that skipped 5 lines and then perform the action. And the action is to insert the line and then we have inserted the line it says break at the 6th line, so after the first time it is the 6th line. So, before the 1st line before the 6th line before the 11th line we are printing a break.

(Refer Slide Time: 36:19)



```
gphani@icme:~/scripts$ cat sample.txt
L1 linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice writer in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$ sed -e '/microsoft/c -----censored-----' sample.txt
```

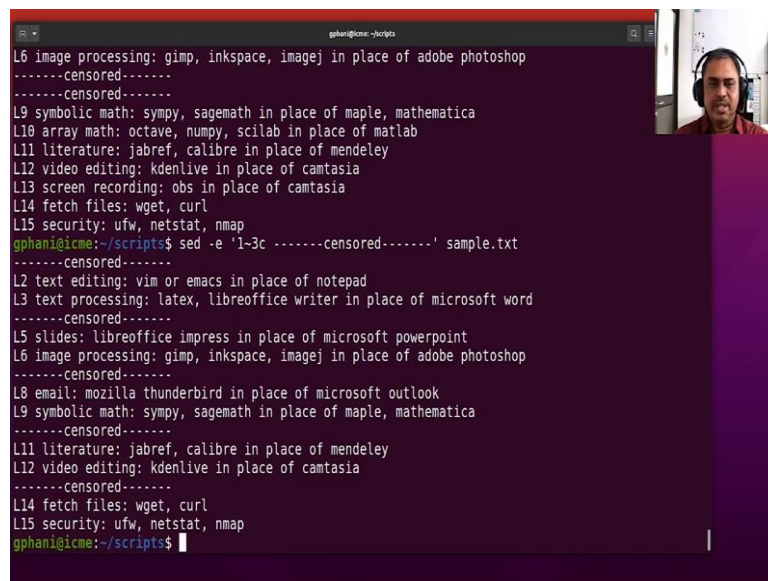
```
gphani@icme: ~/scripts
gphani@icme:~/scripts$ sed -e '/microsoft/c -----censored-----' sample.txt
L1 linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
-----censored-----
-----censored-----
-----censored-----
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
-----censored-----
-----censored-----
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$
```

Now, apart from inserting and appending you also have an ability to change a line that is to replace an entire line with something else. Let us say look at this text, it contains microsoft as one of the words that is listed. So, let us say I do not want to display that on my screen. So, in every line, which contains that particular word, I would like to change it to something else.

So, let us try that out here. So, what are we doing here? So, every line which matches the pattern microsoft, c command is to be executed, c is basically to change, change the line to what is being provided on the command as an option here. So, that particular text which is written here is the new line that has to be replaced with and if the line does not match the pattern of microsoft then the line will be printed as it is.

So, you could see that all those lines which contained the word Microsoft have been replaced with the line which says censored. So, you could actually achieve the replacement of lines by changing it using the c command also. So, naturally the c command can also be used using address ranges.

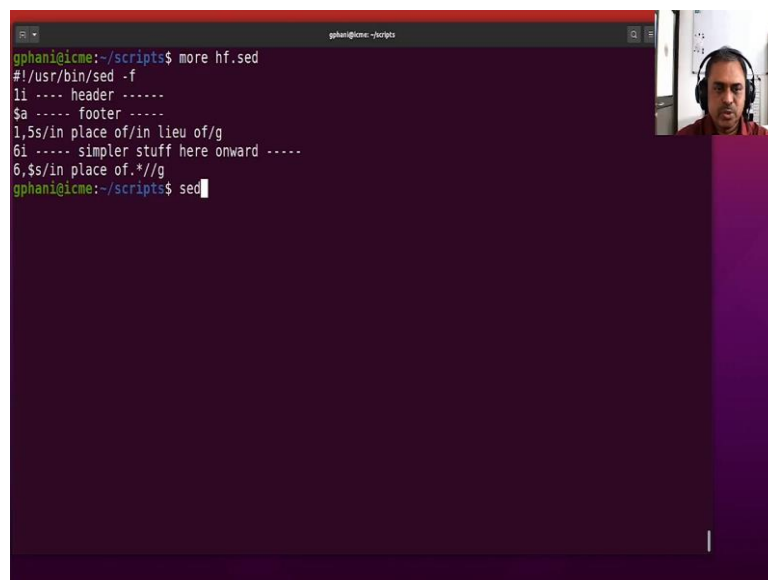
(Refer Slide Time: 37:44)



```
gphani@icme: ~/scripts
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
-----censored-----
-----censored-----
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$ sed -e '1-3c -----censored-----' sample.txt
-----censored-----
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice writer in place of microsoft word
-----censored-----
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
-----censored-----
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
-----censored-----
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
-----censored-----
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$
```

So, for example, I could see like this from the first line onwards, every third line I would like to censored. And you could see that first line onwards every third line has been censored and we are replacing the line with the text that we are providing here in the command.

(Refer Slide Time: 38:06)



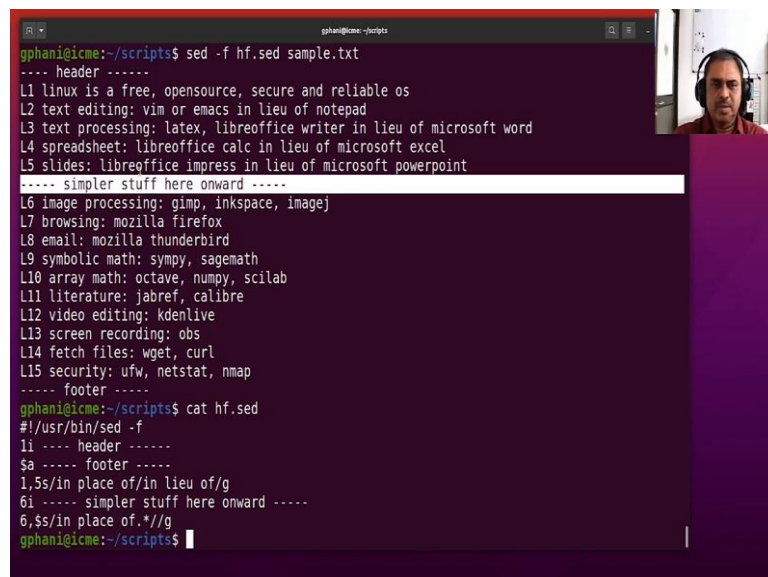
```
gphani@icme:~/scripts$ more hf.sed
#!/usr/bin/sed -f
1i ---- header -----
$a ---- footer -----
1,5s/in place of/in lieu of/g
6i ---- simpler stuff here onward ----
6,$s/in place of.*/g
gphani@icme:~/scripts$ sed
```

Now, you have seen that we are providing the script in the command line itself we could do that in a file also. So, let us go ahead and edit a file that contains the sed script. So, here is an example script where we are mentioning the interpreter which is sed and we are doing certain actions, what we are doing, multiple actions.

So, on the first line if the line number matches one, then insert a header if the line number matches the last one then append a footer. And then from the first line onwards to the 5th line substitute the occurrence of a phrase is in place of bit in lieu of, and then on the 6th line insert a line that is before the 6th line insert a line, which says a phrase. And then from the 6th line onwards till the last line, if you have in place off, then remove the text from that until the end of the line.

So, you can see that we have got about five different commands that are being included in this particular script. And we can execute this script on the sample dot txt file. So, that for every line, all these five commands will be looked at and whichever command is applicable based upon the address range that is specified at the beginning of the line, that particular action will be performed. So, let us go ahead and try this out.

(Refer Slide Time: 39:42)

A terminal window with a dark purple background. The prompt is 'gphani@icme: ~/scripts'. The first command is 'sed -f hf.sed sample.txt'. The output shows a list of 15 lines, each with a line number and a substitution command. Line 1 has 'header' inserted. Line 15 has 'footer' appended. Lines 1-5 have 'in lieu of' substituted for 'is in place of'. Lines 6-15 have the rest of the line removed. The second command is 'cat hf.sed', which shows the script content: a header insertion at line 1, a footer append at line 15, and substitution rules for lines 1-5 and 6-15.

```
gphani@icme:~/scripts$ sed -f hf.sed sample.txt
---- header -----
L1 linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in lieu of notepad
L3 text processing: latex, libreoffice writer in lieu of microsoft word
L4 spreadsheet: libreoffice calc in lieu of microsoft excel
L5 slides: libreoffice impress in lieu of microsoft powerpoint
----- simpler stuff here onward -----
L6 image processing: gimp, inkspace, imagej
L7 browsing: mozilla firefox
L8 email: mozilla thunderbird
L9 symbolic math: sympy, sagemath
L10 array math: octave, numpy, scilab
L11 literature: jabref, calibre
L12 video editing: kdenlive
L13 screen recording: obs
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
----- footer -----
gphani@icme:~/scripts$ cat hf.sed
#!/usr/bin/sed -f
1i ---- header -----
15a ---- footer -----
1,5s/in place of/in lieu of/g
6i ----- simpler stuff here onward -----
6,$s/in place of.*/g
gphani@icme:~/scripts$
```

So, what we are doing here is saying that the sed will use the script coming from a file rather than on the command line and it should execute on a file containing the input stream namely sample dot txt. So, what follows after minus f is interpreted as the file that contains the sed commands. So, now, you can see that the actions that we mentioned are here, so let me just output the set comments also here.

So, the first line, the header has been inserted before it. So, you could see that it has occurred here. And then the last line, a footer line has been appended. So, you could see that after the 15th line, you have got the footer appended. And then from the 1st to the 5th lines, the phrase in place of is replaced with in lieu of. So, you could see here, it is all in lieu of is not in place of.

And from the 6th onwards, what has happened is, we have actually replaced the text that follows in place of and then alternative commands that are listed on the line all of that has been removed. And before the 6th line, a line has been inserted, it is as simple stuff here. So, that has been inserted here. So, you could see that a number of actions can be listed and all those actions will be performed on all the lines of the input stream provided the address ranges matching and address range can be given with respect to the digits or by pattern.

(Refer Slide Time: 41:12)

```

gphani@icme:~/scripts$ more clean.sed
/[[:alpha:]]{2}[[:digit:]]{2}[[:alpha:]]{2}[[:digit:]]{2}/!d
s/[ ]+//g
s/([[:digit:]]+).*/\1/g
gphani@icme:~/scripts$ more block-ex-6.input
This file contains comments and numbers
Lines containing roll number and the fee paid need to be picked up
mm22b901 21000
Sum of the total fee paid needs to be printed at the end
Also a list of students, who paid how much
mm22b902 21000
Input text may contain some comments in between
mm22b906 21800
mm22b903 21500 paid through DD
mm22b905 22000
updated as on Jan 26, 2022 by Phani
gphani@icme:~/scripts$ sed -E -f clean.sed block-ex-6.input
mm22b901 21000
mm22b902 21000
mm22b906 21800
mm22b903 21500
mm22b905 22000
gphani@icme:~/scripts$

```

So, you could think of multiple actions that you could perform using the sed scripts rather than writing all those commands in the command line itself. So, let us say we would like to use this to clean up the input file which we used in the Auc session where we had some roll numbers and the amount of fees paid.

So, here is an example of the input which we have used in the Auc session, where the text is gobbled up with other information that we wanted to have, namely the roll number and the fees paid. So, we would like to remove those lines which do not contain the roll number pattern followed by a integer which contains a fees.

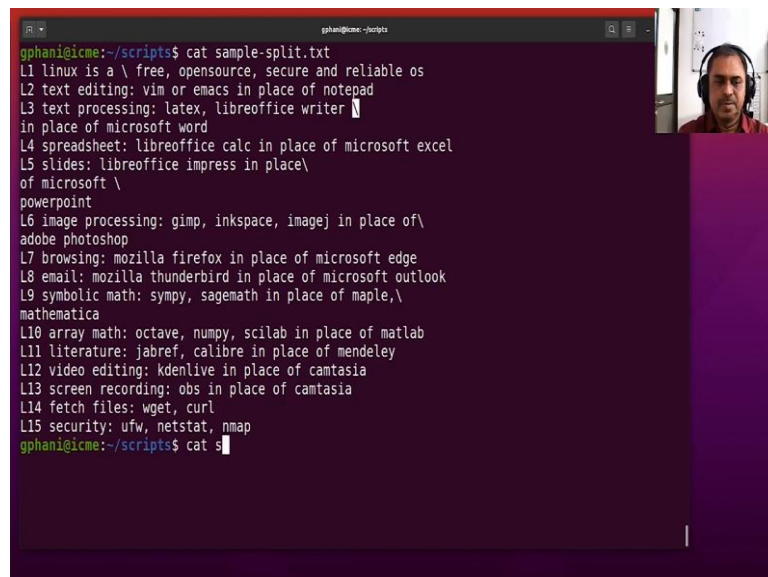
So, what we are doing here is this action says that all those lines on which a pattern that looks like a roll number is to be found. Except those lines, the ampersand says that except those lines which contain the roll number delete, which means that only the lines which contain the roll number pattern only should be printed. Now, that line, the next action will be followed.

The next action is if there are multiple spaces, remove them and substitute with the single space. So, that is done. After that, what is done is if you have anything after which the number has come, then only the number has to be kept. And remaining space in front of the numbers could be replaced. So, what will be the output of this is something that you could see here.

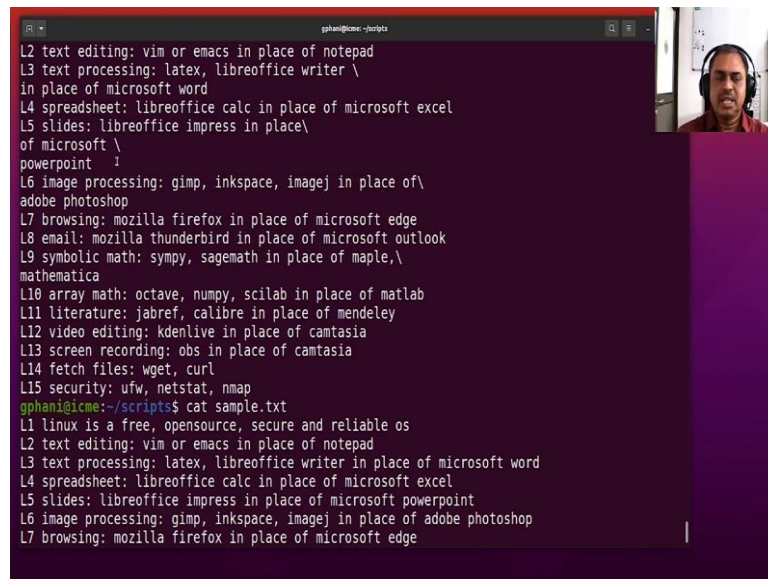
I am going to use minus capital E option because the ER E has to be used for this complicated pattern we have used here, extended regular set has to be used. And minus f because we are going to read the comments from the file and block. Example 6 dot input is the file in which the text is there. So, when we run here, you can see that the output that has come is very clean.

It has only the information that we were seeking, and not the other things that are in the file. So, you could also imagine that the output of this particular sed command, as listed here, is now more ready for further processing by another script, which it does not have to worry about the presence of the remaining text. So, you could use sed to clean up a text to prepare it for further processing by another script.

(Refer Slide Time: 43:47)



```
gphani@icme: ~/scripts
gphani@icme:~/scripts$ cat sample-split.txt
L1 linux is a \ free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice writer
in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place\
of microsoft \
powerpoint
L6 image processing: gimp, inkspace, imagej in place of\
adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple,\
mathematica
L10 array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$ cat s
```

A terminal window with a dark purple background. It contains a list of software alternatives, each preceded by a label (L2 to L15). The list includes text editors, text processors, spreadsheets, slides, image processors, browsers, email clients, symbolic math, array math, literature, video editing, screen recording, file fetching, and security tools. The terminal prompt is 'gphani@icme: ~/scripts'. A video call inset in the top right corner shows a man with a beard and headphones. The list of alternatives is as follows:

```
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice writer \
in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place\
of microsoft \
powerpoint \
L6 image processing: gimp, inkspace, imagej in place of\
adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple,\
mathematica
L10 array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$ cat sample.txt
L1 linux is a free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice writer in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
```

Now, the last command that we would like to mention is to join lines. Joining lines requires the set to read one more line into the buffer and this is achieved in a limited number of commands. And let us see how this can be used to join some lines which fit a particular pattern. Now, sed commands can be constructed as loops and can perform complicated actions.

So, to illustrate that I will use a simple task here. The file that we are showing here is having backslash characters, there is one backslash character within the text but all others are occurring exactly at the end of the line. The purpose is that we would like to break the lines using backslash to ensure that the width of the line is within a particular limit, but for the processing, I would like to join the lines, so that I have the same output as what I have shown you in sample dot txt.

So, here some of the lines are very long, but here I have broken them using backslash So, I would like to join the lines which are ending with a backslash. Now, you can see that I have broken the 5th line into three parts, which means that after I join the 2nd part, I should continue to join with a 3rd part also. And when does it stop only when there is no backslash at the end of the line. So, this can be achieved by using a loop.

(Refer Slide Time: 45:16)

```
#!/usr/bin/sed -f
:x /\$/N
/\$/s/\\n//g
/\$/bx

gphani@icme:~/scripts$ sed -f join.sed sample-split.txt
L1 linux is a \ free, opensource, secure and reliable os
L2 text editing: vim or emacs in place of notepad
L3 text processing: latex, libreoffice writer in place of microsoft word
L4 spreadsheet: libreoffice calc in place of microsoft excel
L5 slides: libreoffice impress in place of microsoft powerpoint
L6 image processing: gimp, inkspace, imagej in place of adobe photoshop
L7 browsing: mozilla firefox in place of microsoft edge
L8 email: mozilla thunderbird in place of microsoft outlook
L9 symbolic math: sympy, sagemath in place of maple, mathematica
L10 array math: octave, numpy, scilab in place of matlab
L11 literature: jabref, calibre in place of mendeley
L12 video editing: kdenlive in place of camtasia
L13 screen recording: obs in place of camtasia
L14 fetch files: wget, curl
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$ sed --debug -f join.sed sample-split.txt

L12 video editing: kdenlive in place of camtasia
INPUT: 'sample-split.txt' line 18
PATTERN: L13 screen recording: obs in place of camtasia
COMMAND: :x
COMMAND: /\$/ N
COMMAND: /\$/ s/\\n//g
COMMAND: /\$/ b x
END-OF-CYCLE:
L13 screen recording: obs in place of camtasia
INPUT: 'sample-split.txt' line 19
PATTERN: L14 fetch files: wget, curl
COMMAND: :x
COMMAND: /\$/ N
COMMAND: /\$/ s/\\n//g
COMMAND: /\$/ b x
END-OF-CYCLE:
L14 fetch files: wget, curl
INPUT: 'sample-split.txt' line 20
PATTERN: L15 security: ufw, netstat, nmap
COMMAND: :x
COMMAND: /\$/ N
COMMAND: /\$/ s/\\n//g
COMMAND: /\$/ b x
END-OF-CYCLE:
L15 security: ufw, netstat, nmap
gphani@icme:~/scripts$
```

So, the loop is constructed in this fashion. So, here you see that every of these lines will be executed for every line that is being read into the buffer and there is a label, colon indicates that there is a label. So, that label is ignored while execute but it is only used for branching. So, the first line what it does is whenever they say backslash followed by the end of the line, then it will read the next line in the buffer.

So, that it is basically like is joining the line following the one which is being processed. The next command what it does is on the lines which have a backslash. If there is a character, new line after the backslash it will be replaced with null. So, it is basically to clean up the backslash character. And what this does is on those lines, which contain backslash at the end of the line, we branched the first line.

So, when we branched the first line, we actually again, do the same action namely to join with the next line. So, when do we not branch and go on to the next cycle is when the line does not contain a backslash at the end of the line, which means that a recursive action is performed. So, one has to watch out that this recursive loop could get into an infinite loop if you do not design these patterns correctly.

But in our case, we have already verified and I will now illustrate how does that work. So, you can see that now the joining has happened quite nicely as expected. And our script has not bothered to replace the backslash within the text because it is not occurring immediately before the end of the line.

So, here is an example of branching and labelling of lines so that a loop can be constructed within the script. And in that sense, a set is a fairly well-developed programming language. But since it operates only on the text scripts, it may not be a good idea for you to play around with it.

Because some of the times if you do not watch out, you may end up in infinite loop. But luckily, there is a debug option available. So, if you wish to enable that, then what happens is that for every line that is being processed, sed will tell you what was the line that was read, what was a pattern that was used, what are the comments that are executed and when does the cycle get over.