

**System Commands**  
**Professor Gandham Phanikumar**  
**Metallurgical and Materials Engineering**  
**Indian Institute of Technology, Madras**  
**Command line editors - Part 01**

(Refer Slide Time: 0:14)

## Command line editors



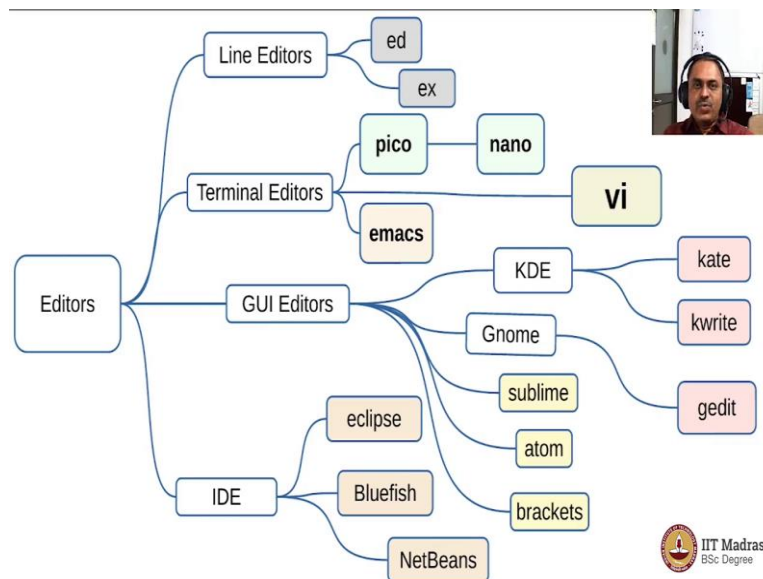
working with text files in the terminal



Welcome to the session on command line editors. In this session, we will work with text files. When we have situations like we have logged into a supercomputer or a remote computer with the poor bandwidth, we often have a requirement to be able to edit some files for submitting our job for execution.

And within a terminal to be able to do that editing is going to be very convenient, because often the file transfer will also take more than one hop to go into a system such as well within the network of an organization where such facilities are made available to you. So, the ability to use a command line editor is going to be important for you to make the best use of such an environment.

(Refer Slide Time: 1:08)



So, when it comes to editors, we have various classes of editors. So, we have got this so-called line editors, that is editors which work in a terminal environment, and help you edit one line at a time. And the old editors such as the ed 'ed' or ex, the ex-editor, which is slightly improved version of the ed editor. These come from the days of Unix. And they are almost always there for any flavour of Unix or Linux.

And terminal editors, which are visual in nature are more popular because of the experience of the user in editing the file. There is an editor which came along with a popular email application called pine, which is called pico. And it could also be launched as a standalone editor. And additional features were added to this editor and it is now called as a nano editor. It is one of the most popular, simple and yet very user-friendly text editor within the terminal environment that we can find today.

The vi editor is perhaps the most popular and also the most complex editor for the terminal environment. And emacs is an alternative to the vi editor. One normally says that among programmers, there are two kinds, the ones who are fanatical about vi editor and the rest who are fanatical about emacs. It is rare to find someone who likes both. And as for me, I am a vi editor fan.

There are graphical user interface editors also available and these are different for different user interfaces such as the KDE environment would have kate or kwrite, which are like the notepad of Windows operating system and in the Gnome environment, it is the gedit which would provide you a GUI editor for pure text files.

And there are editors which open a window in the GUI environment and allow you to edit files with lots of features. And for programmers, these are quite popular tools such as sublime, or atom; atom editor is quite popular among the GitHub users. And then an editor called brackets which is also popular among those who write HTML code.

And there are so called the integrated development environments which come with an editor, compiler, debugger and so on. And we have the popular IDEs on Linux such as Eclipse, Bluefish or NetBeans from Apache. So, while you have a very wide range of editors that are there, it is very useful for us to know one or two editors that would work without any additional installation that is required. And in that sense, vi editor and nano editor would fit the bill because these come automatically bundled with every Linux operating system.

(Refer Slide Time: 4:16)

## Features

- Scrolling, view modes, current position in file
- Navigation (char, word, line, pattern)
- Insert, Replace, Delete
- Cut-Copy-Paste
- Search-Replace
- Language-aware syntax highlighting
- Key-maps, init scripts, macros
- Plugins



What are the kinds of features that you must keep in mind while selecting an editor? You must be able to scroll through the file, you must be able to change the way you view the file that is page by page, to be able to look at different portions of the text file that you are editing. You should also be able to see where are you in the current position, which line number you are within the file and so on.


Navigating within that text file, either character by character or word by word or line by line or by matching a pattern using regular expressions would be very useful when you are looking for a particular keyword that you would like to change and perform some code editing.

The requirement to insert text, replace some text or delete some text is a very common requirement for all of us. Cut copy paste is also required for us because we are going to move text around within our files. We should be able to search and replace a portion of the text, either in the entire text file or in a portion of the text file. It would be good if the editor is language aware so that it can provide us a syntax highlighting so that minor syntactical errors can be avoided while we write the code in the editor itself.


And the added benefits would be when we are able to do some key mappings so that certain combinations of keys can perform slightly elaborate operations within the editor. And if the editor can start with the init script, where such key mappings could be made available, then that would be good. And if you are able to record a macro and then play it so that we perform some repetitive actions on multiple text files, that also would be an added benefit.

And the capabilities of the editor can be extended by plugins. And so those editors that support the concept of plugins, would also be a good choice. And as you could expect, both a vi editor as well as the emacs editor would satisfy all these requirements.

(Refer Slide Time: 6:26)



ed	
Show the Prompt	P
Command Format	[addr[,addr]]cmd[params]
commands for location	2 . \$ % + - , ; /RE/
commands for editing	f p a c d i j s m u
execute a shell command	!command
edit a file	e filename
read file contents into buffer	r filename
read command output into buffer	r !command
write buffer to filename	w filename
quit	q



So, we will actually start off learning how to edit line by line using the editor. I do not think you will actually be using it in your daily life. But the commands that you will be learning through this will be used as part of vi editor and therefore it is worth seeing from where these commands have originated.

And also, in an emergency situation where there is absolutely no editor, then ed will always be there with every Unix or Linux variant. And therefore, you still at least know how to go

about editing some files if other tools are not available. Here is a list of features for the ed editor, which is same as for the ex-editor, except for the ex-editor is slightly more user friendly. So, by default, it has no prompt. And if you press the capital P the prompt will be shown.

And most of the commands have to be given a certain address, or a range of lines on which it has to act. So, the command format would take the starting address, ending address, followed by the command and then parameters for that command to act on. And to locate the cursor on any particular line of the file, you could use the line number itself, like for example, if you press 2, it means that you are in the second line of the text file.

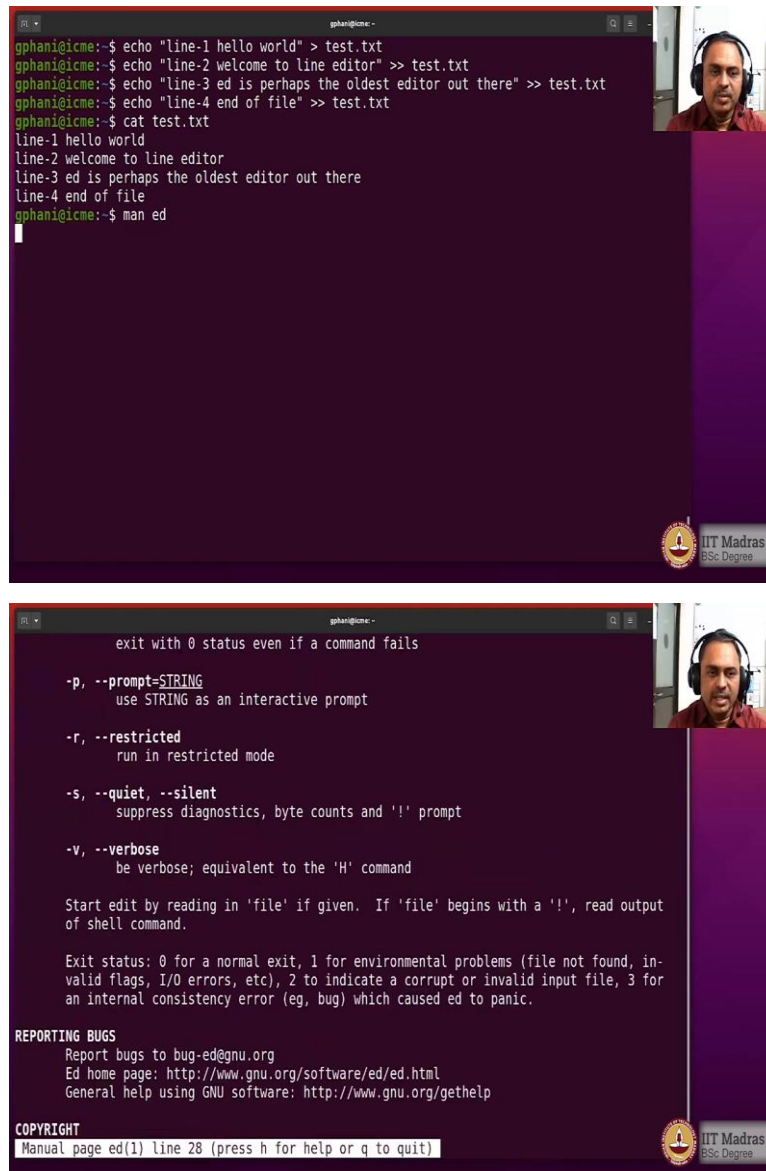
If you press a dot, it means that you are referring to the current client that the cursor is. A dollar would refer to the last line, percentage sign would refer to all the lines so that any action that you are doing will be applicable to all the lines, plus and minus are used for the line that is after the cursor or before the cursor respectively, comma is also going to refer to the entire buffer that is the whole of the text file.

And the semicolon is used to refer from the current position in the text file until the end of the text file. And then a regular expression given with the two forward slashes can also be used to refer to a location where that particular regular expression has been matched. And then there are a bunch of commands for editing. So, these single letters are do very powerful actions and we will come to that when we come to the demo.

You could also execute a command within the ed editor by using the bank in front of the commands that you want to do. So, the command will be passed on to the shell, and it will be executed there and the output will be shown on the screen. And you can edit a particular file by using the file name. Or you could read the contents of any file into the buffer of the file that you are editing currently by using the r command. And you could also read the output of any command into the buffer of the file that you are executing by using the combination of r and the exclamation mark preceding the command.

You could save whatever you have done to the file by using the w command and then once you have saved you can come out of it using the q command. So, ed editor that way is fairly comprehensive when it comes to a simple code editing that is required. But it is quite painful because you do not get to see the whole file and you also cannot move around the file visually. So, it is not a visual editor. It is a line by line editor. So, let us go ahead and explore some of those features using demo.

(Refer Slide Time: 9:43)



```
gphani@icme:~$ echo "line-1 hello world" > test.txt
gphani@icme:~$ echo "line-2 welcome to line editor" >> test.txt
gphani@icme:~$ echo "line-3 ed is perhaps the oldest editor out there" >> test.txt
gphani@icme:~$ echo "line-4 end of file" >> test.txt
gphani@icme:~$ cat test.txt
line-1 hello world
line-2 welcome to line editor
line-3 ed is perhaps the oldest editor out there
line-4 end of file
gphani@icme:~$ man ed

exit with 0 status even if a command fails

-p, --prompt=STRING
    use STRING as an interactive prompt

-r, --restricted
    run in restricted mode

-s, --quiet, --silent
    suppress diagnostics, byte counts and '!' prompt

-v, --verbose
    be verbose; equivalent to the 'H' command

Start edit by reading in 'file' if given. If 'file' begins with a '!', read output
of shell command.

Exit status: 0 for a normal exit, 1 for environmental problems (file not found, in-
valid flags, I/O errors, etc), 2 to indicate a corrupt or invalid input file, 3 for
an internal consistency error (eg, bug) which caused ed to panic.

REPORTING BUGS
Report bugs to bug-ed@gnu.org
Ed home page: http://www.gnu.org/software/ed/ed.html
General help using GNU software: http://www.gnu.org/gethelp

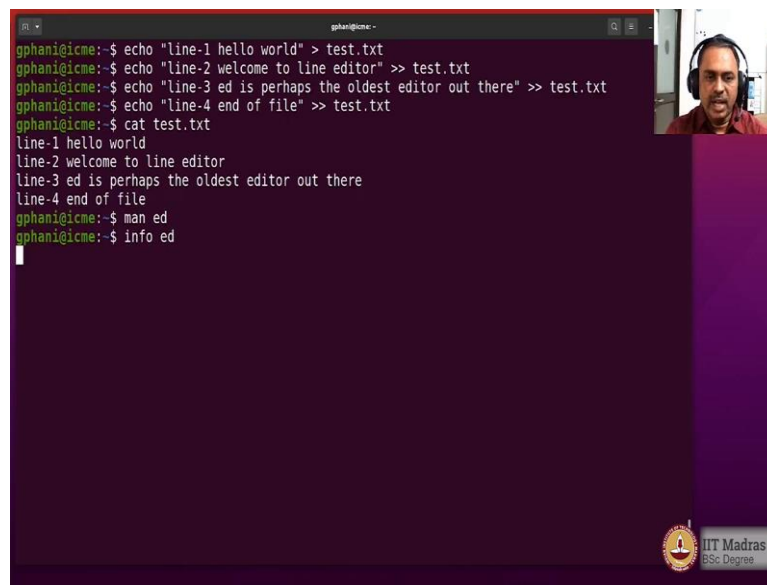
COPYRIGHT
Manual page ed(1) line 28 (press h for help or q to quit)
```

Let us look at the illustration of ed as an editor for line editing. It is solely to learn how these shortcuts for the various actions are coming. It will be also repeated in the vi editor. We need a file to play around so we do not want to touch important files in our home directory. So, we will create a test file that we could edit.

So, I would use the output of echo command along with some line number indicators for us to understand which line we are talking about. So, the very first line would be hello world. And I would like to write this to test dot txt and in the second line, I would like to append it. So, you have got the second line here and there, I would write it as welcome to line editor. And then the third line, I would write something else, ed is perhaps the oldest editor out there. And the fourth line I have end of file.

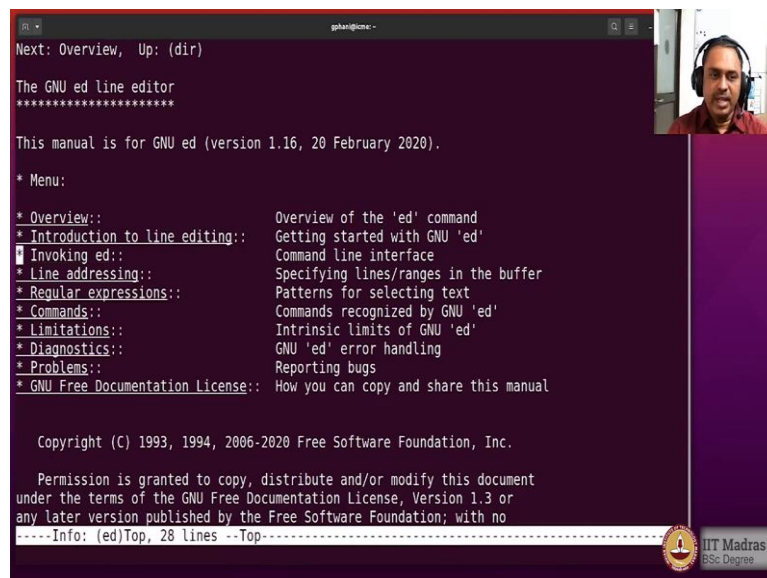
So, look at the file, and you can see that there are four lines now. Now, we would like to edit this file to make some changes. So, that is where we are going to have the ed editor coming up. So, to know more about the ed editor, so you can press man ed, and you see that there is some manual page that is available for the editor, and it does not give you much about the commands etcetera. So, that you could do from info.

(Refer Slide Time: 11:27)



A terminal window with a dark background and light green text. The prompt is 'gphani@icmc: ~'. The user enters several 'echo' commands to create a file 'test.txt' with four lines: 'line-1 hello world', 'line-2 welcome to line editor', 'line-3 ed is perhaps the oldest editor out there', and 'line-4 end of file'. Then, the user enters 'cat test.txt' and sees the four lines. Finally, the user enters 'man ed' and 'info ed', which leads to the manual page shown in the next slide. A small video feed of a man with a headset is visible in the top right corner. An IIT Madras BSc Degree logo is in the bottom right corner.

```
gphani@icmc:~$ echo "line-1 hello world" > test.txt
gphani@icmc:~$ echo "line-2 welcome to line editor" >> test.txt
gphani@icmc:~$ echo "line-3 ed is perhaps the oldest editor out there" >> test.txt
gphani@icmc:~$ echo "line-4 end of file" >> test.txt
gphani@icmc:~$ cat test.txt
line-1 hello world
line-2 welcome to line editor
line-3 ed is perhaps the oldest editor out there
line-4 end of file
gphani@icmc:~$ man ed
gphani@icmc:~$ info ed
```



The same terminal window showing the GNU ed manual page. The text is light green on a dark background. It starts with 'Next: Overview, Up: (dir)' and 'The GNU ed line editor'. It then says 'This manual is for GNU ed (version 1.16, 20 February 2020)'. A menu lists various topics like Overview, Introduction to line editing, Invoking ed, Line addressing, Regular expressions, Commands, Limitations, Diagnostics, Problems, and GNU Free Documentation License. The bottom of the page shows copyright information for the Free Software Foundation, Inc. (1993, 1994, 2006-2020) and a permission statement. A status line at the bottom says '-----Info: (ed)Top, 28 lines --Top-----'. The video feed and IIT Madras logo are still present.

```
Next: Overview, Up: (dir)

The GNU ed line editor
*****

This manual is for GNU ed (version 1.16, 20 February 2020).

* Menu:

* Overview::          Overview of the 'ed' command
* Introduction to line editing:: Getting started with GNU 'ed'
* Invoking ed::       Command line interface
* Line addressing::    Specifying lines/ranges in the buffer
* Regular expressions:: Patterns for selecting text
* Commands::          Commands recognized by GNU 'ed'
* Limitations::        Intrinsic limits of GNU 'ed'
* Diagnostics::        GNU 'ed' error handling
* Problems::           Reporting bugs
* GNU Free Documentation License:: How you can copy and share this manual

Copyright (C) 1993, 1994, 2006-2020 Free Software Foundation, Inc.

Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3 or
any later version published by the Free Software Foundation; with no
-----Info: (ed)Top, 28 lines --Top-----
```



Next: Invoking ed, Prev: Overview, Up: Top

## 2 Introduction to line editing

\*\*\*\*\*


'ed' was created, along with the Unix operating system, by Ken Thompson and Dennis Ritchie. It is the refinement of its more complex, programmable predecessor, 'QED', to which Thompson and Ritchie had already added pattern matching capabilities (\*note Regular expressions:).

For the purposes of this tutorial, a working knowledge of the Unix shell 'sh' and the Unix file system is recommended, since 'ed' is designed to interact closely with them. (\*Note GNU bash manual: (bash)Top, for details about bash).

The principal difference between line editors and display editors is that display editors provide instant feedback to user commands, whereas line editors require sometimes lengthy input before any effects are seen. The advantage of instant feedback, of course, is that if a mistake is made, it can be corrected immediately, before more damage is done. Editing in 'ed' requires more strategy and forethought; but if you are up to the task, it can be quite efficient.

Much of the 'ed' command syntax is shared with other Unix utilities.

-----Info: (ed)Introduction to line editing, 202 lines --Top-----



should be printed, we prefix the command with <, > (comma) which is shorthand for "the whole buffer":

```
*.p
      June 2006
Mo Tu We Th Fr Sa Su
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30
```


Now let's write the buffer contents to a file named 'junk' with the <w> ("write") command:

```
*w junk
137
*
```

Need we say? It's good practice to frequently write the buffer contents, since unwritten changes to the buffer will be lost when we exit 'ed'.

The sample sessions below illustrate some basic concepts of line

-----Info: (ed)Introduction to line editing, 202 lines --40%-----



Next: Line addressing, Prev: Introduction to line editing, Up: Top

## 3 Invoking ed

\*\*\*\*\*

The format for running 'ed' is:

```
ed [OPTIONS] [FILE]
red [OPTIONS] [FILE]
```

FILE specifies the name of a file to read. If FILE is prefixed with a bang (!), then it is interpreted as a shell command. In this case, what is read is the standard output of FILE executed via 'sh (!)'. To read a file whose name begins with a bang, prefix the name with a backslash ('\'). The default filename is set to FILE only if it is not prefixed with a bang.


'ed' supports the following options:

```
.-h
.-help
.-v
.-version
```

Print an informative help message describing the options and exit.

Print the version number of 'ed' on the standard output and exit.

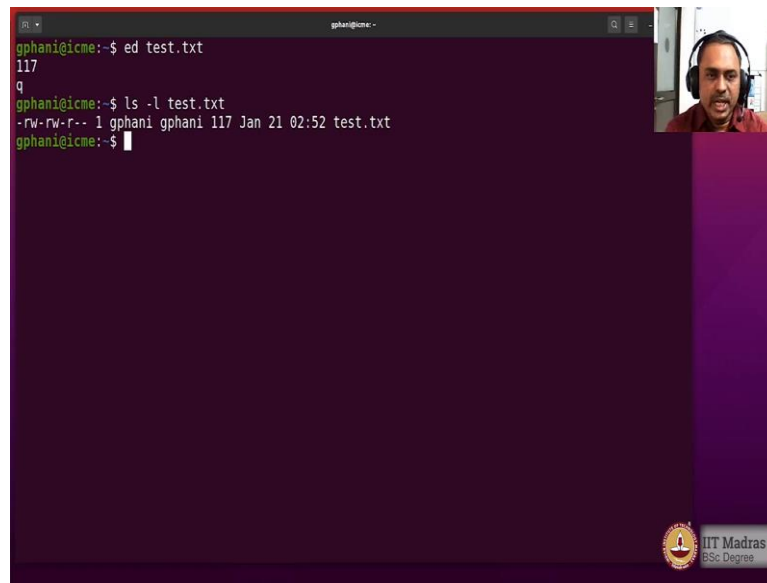
-----Info: (ed)Invoking ed, 66 lines --Top-----





So, info ed would open the info page for the editor, and you can go to the overview, then scroll through and you will go to the second section on Introduction to line editing. And there are some illustrations of how to go about and then you go on to the third session where the format of various commands is discussed and so on, I come out of this help now and we start using the editor.

(Refer Slide Time: 11:56)



```
gphani@icme:~$ ed test.txt
117
q
gphani@icme:~$ ls -l test.txt
-rw-rw-r-- 1 gphani gphani 117 Jan 21 02:52 test.txt
gphani@icme:~$
```

So, the way to invoke it is just ed and then the file name and you see that when it opens. So, a little bit confusing, because we do not know what it is going to do. So, it is actually showing you some number which indicates how many bytes it has read into the memory. So, let us go and confirm that. So, we have to come out of it by using q and ls minus l test dot txt and you will see that 117 is the number of bytes. So, it is reading the entire file into the memory and then it is going to then show you line by line something that you can edit.

(Refer Slide Time: 12:35)

```
gphani@icme:~$ ed test.txt
117
P
*1
line-1 hello world
*$
line-4 end of file
*,p
line-1 hello world
line-2 welcome to line editor
line-3 ed is perhaps the oldest editor out there
line-4 end of file
*2,3p
line-2 welcome to line editor
line-3 ed is perhaps the oldest editor out there
*/hello/
line-1 hello world
*/oldest/
line-3 ed is perhaps the oldest editor out there
*1
line-1 hello world
*+
line-2 welcome to line editor
*
line-1 hello world
*3
line-3 ed is perhaps the oldest editor out there
*;p
```

```
line-1 hello world
*3
line-3 ed is perhaps the oldest editor out there
*;p
line-3 ed is perhaps the oldest editor out there
line-4 end of file
*%p
line-1 hello world
line-2 welcome to line editor
line-3 ed is perhaps the oldest editor out there
line-4 end of file
*,
line-4 end of file
*1
line-1 hello world
*!date
Friday 21 January 2022 02:56:55 AM IST
!
*,p
line-1 hello world
line-2 welcome to line editor
line-3 ed is perhaps the oldest editor out there
line-4 end of file
*4
line-4 end of file
*r !date
39
*
```

```
line-4 end of file
*r !date
39
*,p
line-1 hello world
line-2 welcome to line editor
line-3 ed is perhaps the oldest editor out there
line-4 end of file
Friday 21 January 2022 02:57:15 AM IST
*w
156
*q
gphani@icme:~$
gphani@icme:~$
gphani@icme:~$
gphani@icme:~$
gphani@icme:~$
gphani@icme:~$
gphani@icme:~$ ls -l test.txt
-rw-rw-r-- 1 gphani gphani 156 Jan 21 02:57 test.txt
gphani@icme:~$ cat test.txt
line-1 hello world
line-2 welcome to line editor
line-3 ed is perhaps the oldest editor out there
line-4 end of file
Friday 21 January 2022 02:57:15 AM IST
gphani@icme:~$
```

So, let us go ahead and start again. So, there is no prompt at all and sometimes it is useful to know that there is a prompt for you to type commands. So, press capital P and it will use the prompt which is star you can change it if you wish, but for now, we will go ahead with this. Now, there are some commands that we need to practice.

So, let us say we go to the first line and it shows you what is in the first line. And if you want to go to the last line, plus dollar and you are in the last line. You could actually put a comma and a small p to know what are the contents of the entire buffer. So, all the four lines are shown. You could also give a range. So, you can say 2 to 3 then p. So, the second to third line are getting printed.

So, you can see that the display is actually is scrolling and the star is actually telling you a prompt that you would like to then add some commands after that, but it is not showing you the file in a visual manner. So, that is the whole idea of a line by line editing concept using the line editor ed.

Can we then also look at which line we will be if you were to search for a pattern? So, let us say I search for hello and let us see it is there in the first line. I search for oldest and it is saying that you are in the line 3. So, it would search for the pattern and bring you to the first occurrence of that particular pattern.

And we could also go to a specific line and then when you press a plus it will go to the next line and the plus or minus it go to the previous line. So, you can scroll up and down by looking at one line at a time. You could also go from the specific line to the end of the buffer like this. So, let us say you are going from 3 and from there if you want to go to the end of the end of the buffer. So, you see from the third line till the end of the buffer which is third and fourth lines are being displayed when you use a semicolon followed by p for printing.

Now, percentage is also a short form for the entire set of lines. So, I want to see all of them so I am also able to use the percentage. So, these are all the various ways of locating which line you are going to operate on. So, dot actually says which is the current line right now, so we are actually on the fourth line. And now again, we are back on the first line, so we can actually always see where we are. And therefore, the command that we are going to run is going to be affected on that particular line.

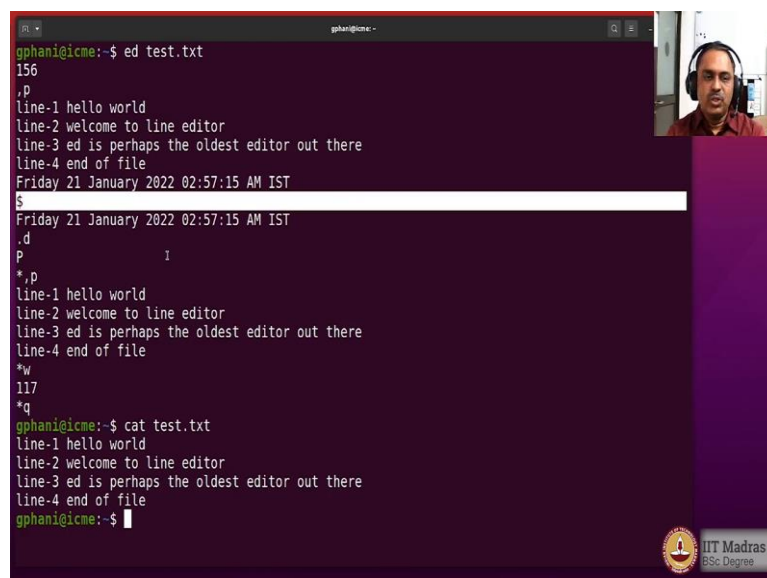
As I mentioned, you could also run some commands here. So, I put a exclamation mark and then put date. So, the command has been run and the display is done. But it does not affect

your buffer yet. So, you could see that the four lines are still there. So, if I want to read the output of the command into the buffer, and I want to specify where I want to read it, let me go to the fourth line. And then here, I want to say read the output of the date command.

So, it comes into the buffer at the bottom, and you can then ask it to show all the lines. So, you can see that it is added at the bottom of the file, but it is not yet written. And I can press w, and then it writes the file. So, now the file is saved. And once you have saved, if you wish, you can come out of the file, so q will get you out of the editor. So, I am back to the prompt. This prompt is something you are already familiar with, this is the prompt of the shell.

And let us look at the occupation of a test dot txt, it is 156 bytes. So, it means it has grown. And sure enough, it is because we have added a line at the end of the file. So, you can see that we actually successfully were able to scroll through the file and add something at the bottom. So, we can try out a few more features now.

(Refer Slide Time: 16:34)



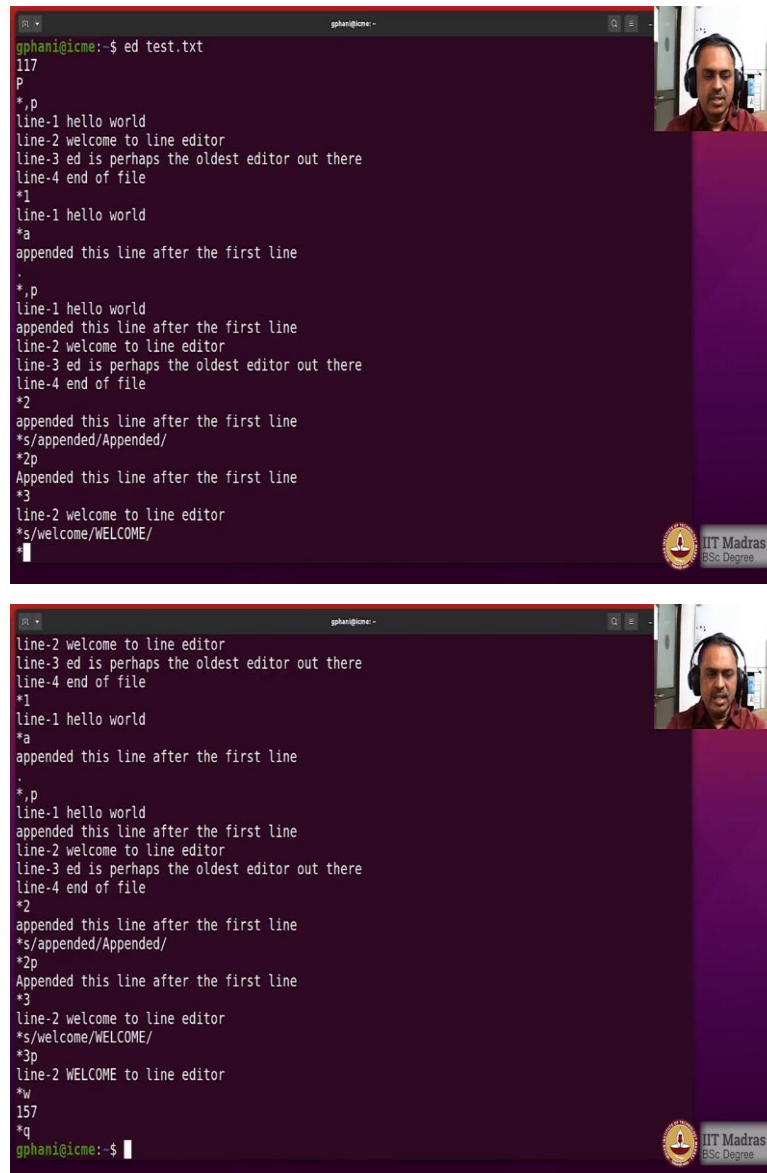
```
gphani@icme:~$ ed test.txt
156
*,p
line-1 hello world
line-2 welcome to line editor
line-3 ed is perhaps the oldest editor out there
line-4 end of file
Friday 21 January 2022 02:57:15 AM IST
$
Friday 21 January 2022 02:57:15 AM IST
*,d
P
*,p
line-1 hello world
line-2 welcome to line editor
line-3 ed is perhaps the oldest editor out there
line-4 end of file
*w
117
*q
gphani@icme:~$ cat test.txt
line-1 hello world
line-2 welcome to line editor
line-3 ed is perhaps the oldest editor out there
line-4 end of file
gphani@icme:~$
```

So, now we have got 5 lines. So, I want to go to the last line. So, I put a dollar. So now, I am in the last line. And I want to say that on the current line, do the action, which is deleting it, so it has done that action. And to help myself I put capital P for the prompt and then I want to see the buffer you will see that the buffer has been reduced, it is only 4 lines now. And if I press w, the file is saved. Come out of it and cat test dot txt and you see that the last line has been deleted.

So, which command has deleted the last line? It is this command dot d that is, in the current position delete the line and what is the current position; dollar says that you are going to the

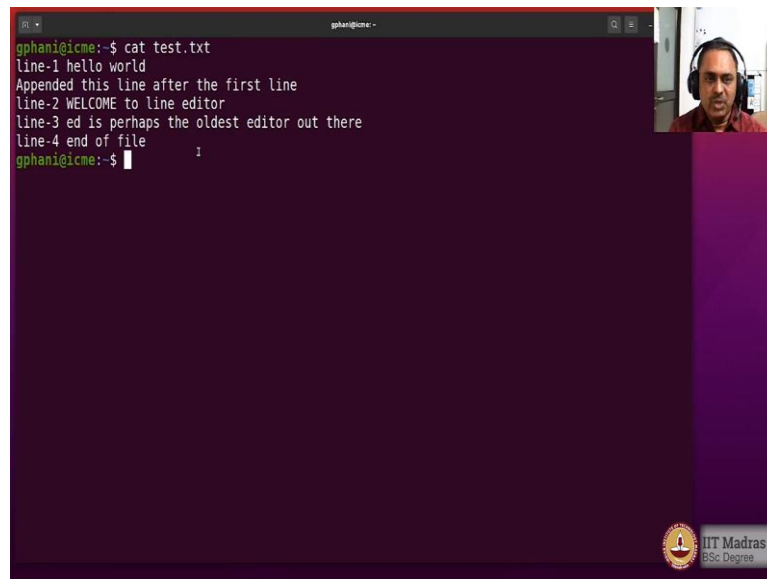
last line of the file and therefore the command together when you read it says that you should delete the last line.

(Refer Slide Time: 17:27)



```
gphani@icme:~$ ed test.txt
117
P
*,p
line-1 hello world
line-2 welcome to line editor
line-3 ed is perhaps the oldest editor out there
line-4 end of file
*1
line-1 hello world
*a
appended this line after the first line
.
*,p
line-1 hello world
appended this line after the first line
line-2 welcome to line editor
line-3 ed is perhaps the oldest editor out there
line-4 end of file
*2
appended this line after the first line
*s/appended/Appended/
*2p
Appended this line after the first line
*3
line-2 welcome to line editor
*s/welcome/WELCOME/
*
gphani@icme:~$
```

```
line-2 welcome to line editor
line-3 ed is perhaps the oldest editor out there
line-4 end of file
*1
line-1 hello world
*a
appended this line after the first line
.
*,p
line-1 hello world
appended this line after the first line
line-2 welcome to line editor
line-3 ed is perhaps the oldest editor out there
line-4 end of file
*2
appended this line after the first line
*s/appended/Appended/
*2p
Appended this line after the first line
*3
line-2 welcome to line editor
*s/welcome/WELCOME/
*3p
line-2 WELCOME to line editor
*w
157
*q
gphani@icme:~$
```



```
gphani@icme:~$ cat test.txt
line-1 hello world
Appended this line after the first line
line-2 WELCOME to line editor
line-3 ed is perhaps the oldest editor out there
line-4 end of file
gphani@icme:~$
```

Now, we can actually go ahead and insert something in between also. So, look at the lines so, I want to go to the first line and there I want to append some text. So, if I want to append some text, I press a and then when I enter it is now asking for the text that I want to happen. So, you can say appended this line after the first line.

Now, once you are done, you should come out of this appending action. So, on an empty line, if you press dot and then enter, then the editor will know that you have finished giving the text and then it will bring the prompt back to you. Now, I go and look at the contents of the file and you can see that what I wrote here is then appended after the first line because my position of the line is known here as 1. So, after that to the appending has taken place and now you can see that the text has been appended after the first line.

Now, we could also do editing of this line. So, this editing can be using some principles of regular expressions that we have learned. So, let me go to the line that we are talking about. So, that is the second line, I want to change the case of the first word. So, what I do is that I search for the word and I replace it with something. So, the second part that comes after the forward slash would be the replacement string. So, I would say replacement is by using the capital A, and I close the path.

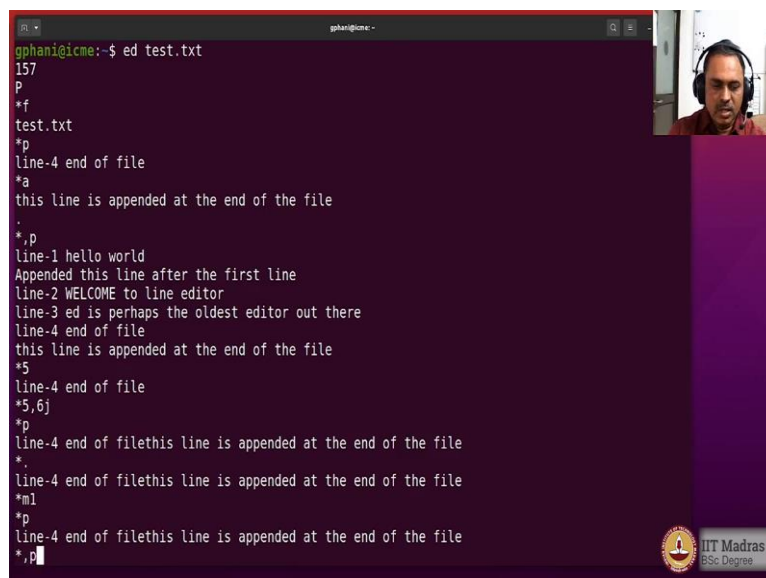
So, now you can see that there are two pairs of regular expressions that are given the first one is for the matching. So, this part is for the matching part and this is further replacement and on which line is going to be active on the line that you have just mentioned. So, we do that and then let us say I want to print the second line. So, you can see that the second line has been modified, the first word has been edited.

So, we successfully edited one line by identifying the line number and doing some action. Now, the editing is done actually by regular expression and replacement but not in a visual sense. So, this is something very different way of editing and this is how people used to do in the distant past.

So, we could also do some more editing. So, let us say we go to the third line, and we want to change something here. So, substitute welcome, and I want to have all capitals, and want to print that line and see that I have changed that line. Now, we have done a couple of changes. So, we go ahead and type the buffer, and then quit, and I clean the screen and cat test dot txt, and you see that we have done the changes.

So, we have edited the file, so that the second line has a capital A. And there is a line also after the first line. And the third line has a capitalized word. So, we have successfully added some text and we have also changed the letters. So, we have been able to edit the file. But of course, it is not very optimal in terms of experience, but in olden days when you had very slow speed of computers, so if you are able to change one line at a time, that is already good enough for you to write a program, because those computers are very slow compared to what we are using today.


(Refer Slide Time: 20:54)




```
gphani@icme:~$ ed test.txt
157
p
*f
test.txt
*p
line-4 end of file
*a
this line is appended at the end of the file
.
*,p
line-1 hello world
Appended this line after the first line
line-2 WELCOME to line editor
line-3 ed is perhaps the oldest editor out there
line-4 end of file
this line is appended at the end of the file
*5
line-4 end of file
*5,6j
*p
line-4 end of filethis line is appended at the end of the file
*
line-4 end of filethis line is appended at the end of the file
*ml
*p
line-4 end of filethis line is appended at the end of the file
*,p
cat test.txt
hello world
WELCOME to line editor
ed is perhaps the oldest editor out there
this line is appended at the end of the file
```



```
gphani@icme: ~  
line-4 end of filethis line is appended at the end of the file  
*,p  
line-1 hello world  
line-4 end of filethis line is appended at the end of the file  
Appended this line after the first line  
line-2 WELCOME to line editor  
line-3 ed is perhaps the oldest editor out there  
*1p  
line-1 hello world  
*2p  
line-4 end of filethis line is appended at the end of the file  
*,p  
line-4 end of filethis line is appended at the end of the file  
*,m0  
*,p  
line-4 end of filethis line is appended at the end of the file  
line-1 hello world  
Appended this line after the first line  
line-2 WELCOME to line editor  
line-3 ed is perhaps the oldest editor out there  
*,u  
*,p  
line-1 hello world  
line-4 end of filethis line is appended at the end of the file  
Appended this line after the first line  
line-2 WELCOME to line editor  
line-3 ed is perhaps the oldest editor out there  
*,w
```



```
gphani@icme: ~  
line-4 end of filethis line is appended at the end of the file  
Appended this line after the first line  
line-2 WELCOME to line editor  
line-3 ed is perhaps the oldest editor out there  
gphani@icme:~$ ed test.txt  
201  
P  
*,p  
line-1 hello world  
line-4 end of filethis line is appended at the end of the file  
Appended this line after the first line  
line-2 WELCOME to line editor  
line-3 ed is perhaps the oldest editor out there  
*%s/(.*/)/PREFIX \1/  
*,p  
PREFIX line-1 hello world  
PREFIX line-4 end of filethis line is appended at the end of the file  
PREFIX Appended this line after the first line  
PREFIX line-2 WELCOME to line editor  
PREFIX line-3 ed is perhaps the oldest editor out there  
*3,5s/PREFIX/prefix/1  
*,p  
PREFIX line-1 hello world  
PREFIX line-4 end of filethis line is appended at the end of the file  
prefix Appended this line after the first line  
prefix line-2 WELCOME to line editor  
prefix line-3 ed is perhaps the oldest editor out there
```



Now, let us look at a couple of more features, so that we understand the commands. So, I put the prompt, and what file am I working on, the f command would show that. So, this command is going to be there also in the vi editor. So, f shows the name of the file that is being edited. And p would show the contents of the current line. So, right now we are at the end of the file, so the last line is being shown.

And a is for appending at the current line. So, I am at the last line. So, I append, and I write something here. So, this line is appended at the end file, and then dot and enter. So, the line is done, you can see all the lines now. So, you could see that I have appended a line at the bottom of the file. Now you could also do some joining here. So, I would go and try to join the last line to the forth line 4.

So, I go to 5, line 4, and the syntax would be I want to join 5 and 6 with j. And then you can look at the current line. And you can see that these last two lines have been joined. So, line for end of file, enter without anything between that and the next line, this line is appended at the end of the file. So, you can see that the two lines have been joined. So, 5 comma 6 j means join the fifth and the sixth lines.

Now, you could also move a particular line to a particular position. So right now, where are we? So, you can see that we added this line. So, we can say move this line to the first position and then let us go and see what happened. So, you can see that the last line has come just below the first line. So, you could actually now print the second line, and you see that it has come. And which means that I have successfully been able to move the last line to that.

Now, if I want to make this as a first line so first, let me check am in the line 2. So, I can say move this line to the 0 position and then I can ask it to print all the lines. And you can see that that particular line I am working on line 4 end of file, etcetera, is now at the first position. So, m0 would mean go to the very beginning of the file, m1 means put it after the past line, and so on. So, you could actually give a number after m to say to which line you are to move the current line. And that way you can move the lines also, up and down.

And whatever we have done, we do not want that change. So, I want to undo so I press u and press an Enter and you can see again all the lines and see that the change I have made is reverted back. So, you can see that here, the line 1 and the line 4 are one after the other, then when I moved, then I have got the line 4 and line 1 are like this, and then after I did undo, then you can see that they are back to the original form. So, you can also do some reversal of the actions that you have taken.

And at any time, you want to read the file to the disk, and then you can come out. And cat, to see that whatever changes you have done, habit affected the file. This is how the edit that is actually going to help people to go around changing the text. Now, we will just illustrate one more feature of the ed editor where the regular expression power can be illustrated.

So, look at all the lines. So, let us say I would like to move all the lines a bit side by putting something in the front. So, what I would do is I should say the entire range I want to apply so percentage will say that all the lines are the address range. And I want to search for something that is basically any character that can be matched.

So, now, you will see that we have grouped dot star, which means that all the characters that can be matched are grouped like that and I want to replace them by writing something in front and what do I want to write in the front, I would say PREFIX and then a space and then backslash 1. Backslash 1 means whatever we have grouped just now is going to be printed in that position.

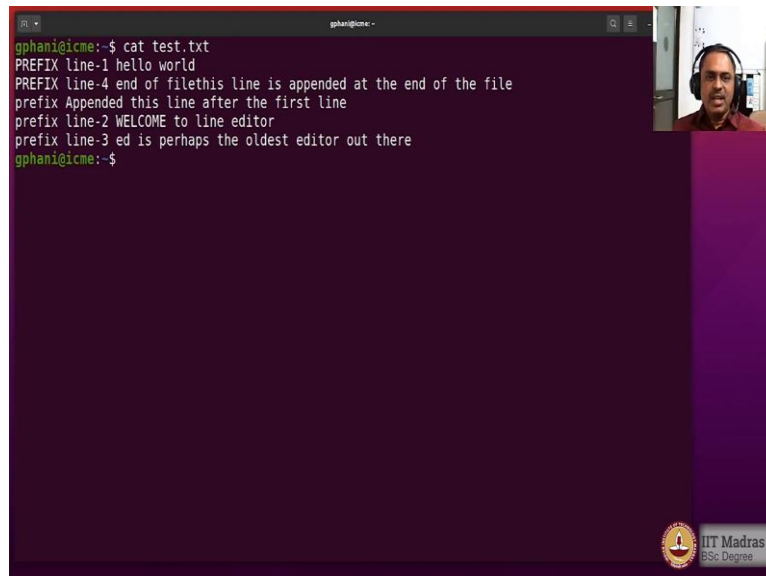
So, essentially, what we are expecting is every line will be now prefixed with the word PREFIX and a space and I press a forward slash to complete the replacement expression and an enter. So, now, this action has been taken place on all the lines, and I can now print all the lines to see what happened. So, PREFIX word has come in front of all the lines. So, you can see that the regular expressions that we have learned in the previous class, and also the back substitution can also be used for editing action straightaway and it will apply to all the lines are any specific line.

So, let us say I want to reverse some of it and make it applicable only for specific lines. So, that I would do. Let us say from the third to fifth lines, I want to substitute PREFIX word with a smaller version. And now you can see that the text has been modified. So, from the third onwards to fifth so, third, fourth fifth lines have changed the first word were with a regular expression, I am substituting capital PREFIX with a small prefix.

So, you can see that we are actually using the regular expressions and replacements to change one word at a time in a given line and we can move the lines up and down, we can add a line, we can delete a line and we can also append something at the bottom of the file and we can also put some things in front of the line or at the end of the line using the regular expressions and therefore, we are able to edit the lines.

So, this is a very powerful feature of the ed editor. And it is pretty good for us for daily life editing of source code, but today we have a much more powerful and visual experience with the vi editor. So, ed editors that is why is not so popular today. But it continues to be there in the system package just so that you can always try it out if you are interested in some historically useful editors that people have used in the ester years.

(Refer Slide Time: 27:24)



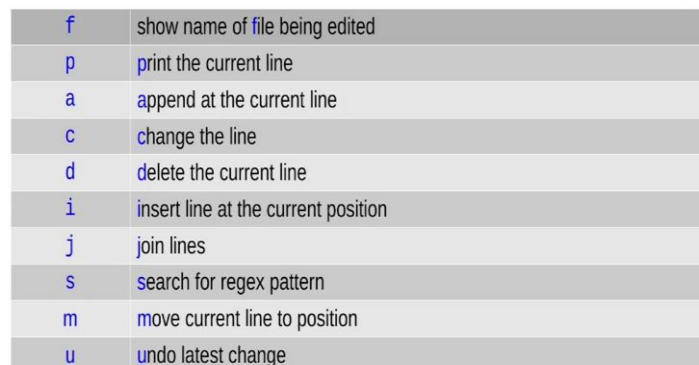
```
gphani@icme:~$ cat test.txt
PREFIX line-1 hello world
PREFIX line-4 end of filethis line is appended at the end of the file
prefix Appended this line after the first line
prefix line-2 WELCOME to line editor
prefix line-3 ed is perhaps the oldest editor out there
gphani@icme:~$
```

The terminal window shows the output of the 'cat test.txt' command. The file contains five lines of text, some with prefixes and some with lowercase prefixes. The terminal window also shows a small video feed of the presenter in the top right corner and the IIT Madras logo in the bottom right corner.

So, let me write the buffer and cube okay and cat, what did we do the test you can see that we have edited the file significantly all using the line editor called ed.

(Refer Slide Time: 27:33)

## ed / ex commands



<b>f</b>	show name of <b>f</b> ile being edited
<b>p</b>	<b>p</b> rint the current line
<b>a</b>	<b>a</b> ppend at the current line
<b>c</b>	<b>c</b> hange the line
<b>d</b>	<b>d</b> eleate the current line
<b>i</b>	<b>i</b> nsert line at the current position
<b>j</b>	<b>j</b> oin lines
<b>s</b>	<b>s</b> earch for regex pattern
<b>m</b>	<b>m</b> ove current line to position
<b>u</b>	<b>u</b> ndo latest change

The table lists ten commands for the ed and ex editors, each with a one-letter command and its description. The commands are: f (show name of file being edited), p (print the current line), a (append at the current line), c (change the line), d (delete the current line), i (insert line at the current position), j (join lines), s (search for regex pattern), m (move current line to position), and u (undo latest change). The commands are listed in a table with a light gray background and a dark gray border. The table is located on the left side of the slide, and the video feed of the presenter is on the right side.



Here is a summary of commands that we have explored in the demonstration for both the editors ed as well as example. And each of these commands is a one letter command. And these letters are the same in vi editor also. So, it would be good to also associate this one letter commands with their meaning and I have highlighted them in blue color to hint that one can actually try to remember these already.