The vi Editor

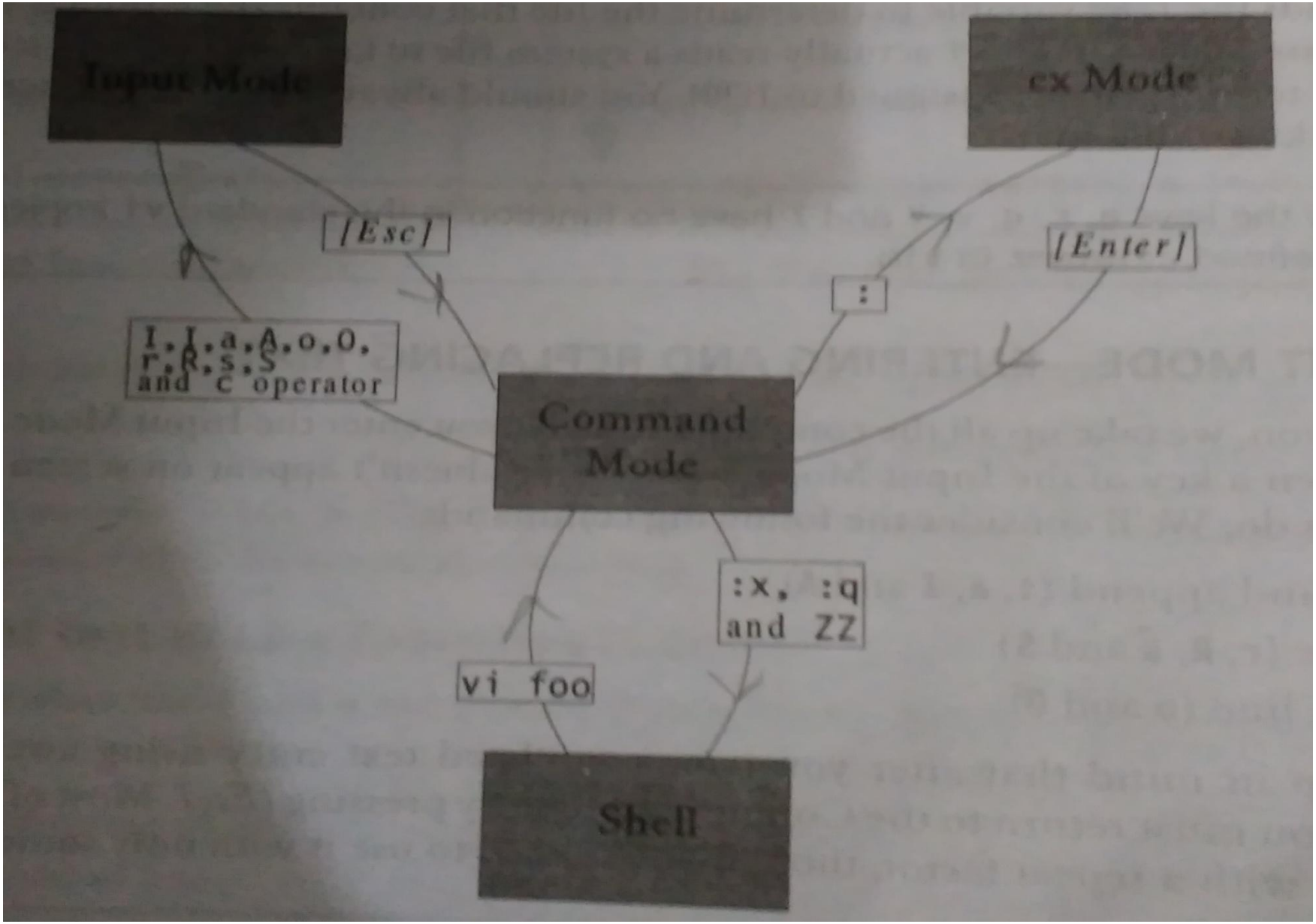
vi Editor

- Operates in three modes

-> **input mode** is used to enter text

-> **ex mode**(or last line mode) is used for file handling and substitution

-> **command mode** is used to enter commands that operate on text or control cursor motion


Input Mode
ex Mode
[Esc]
[Enter]
:
I, I, a, A, o, O,
r, R, s, S
and c operator
Command Mode
:x, :q
and ZZ
vi foo
Shell

INPUT MODE

- The following commands

-> Insert and append (i, a, I and A)

-> Replace (r, R, s and S)

-> Open a line (o and O)

Insertion of Text(i and a)

i

Pressing this key changes the mode from Command to input

- If the **i** command is invoked with the cursor positioned on existing text, text on its right will be shifted further without being overwritten

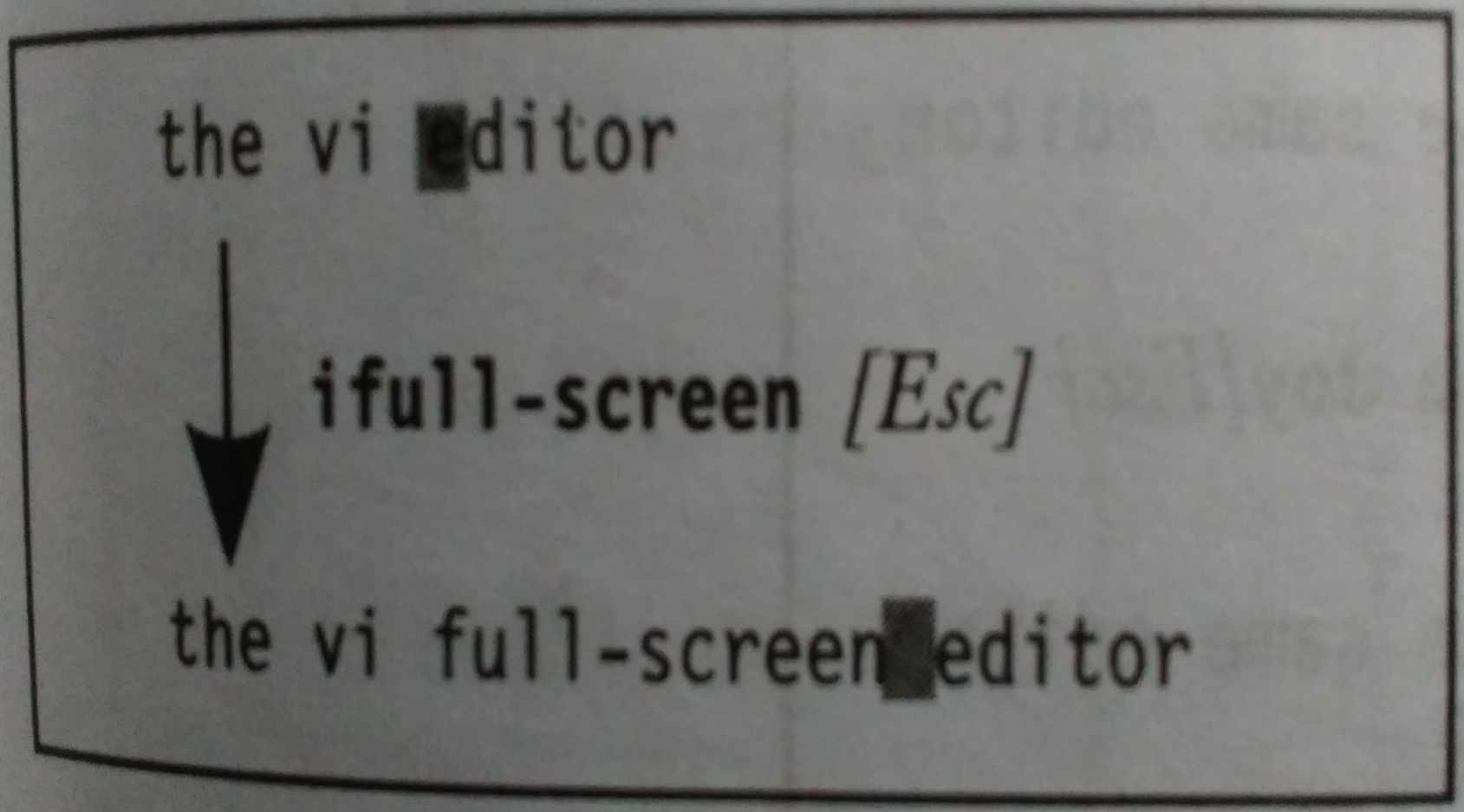


Fig. 7.3 Text Insertion with i

- Other methods of inputting text. To append text to the right of the cursor position use

  a

  followed by the text you wish to type. After finish editing , press [ESC]

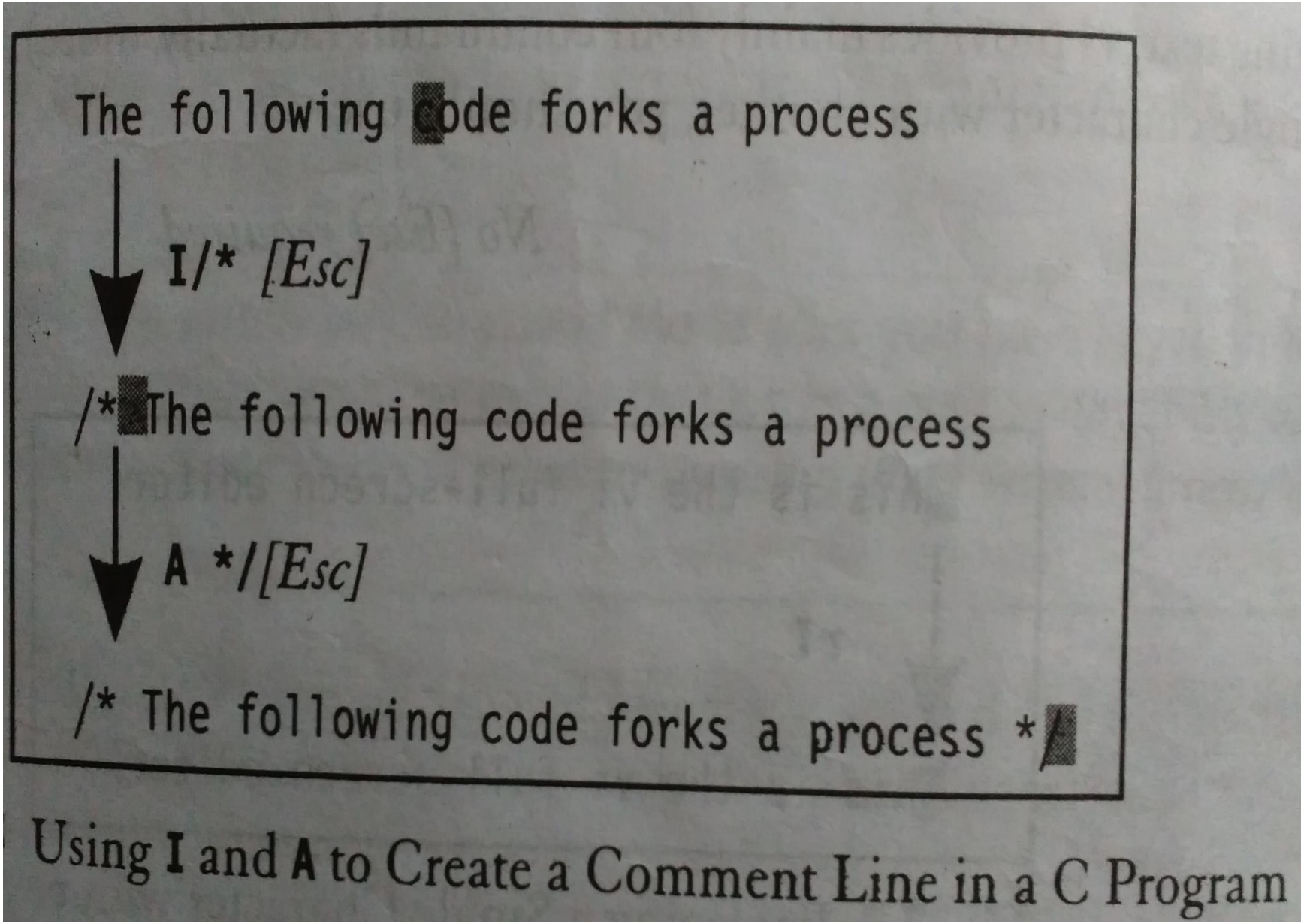
the vi full-screen editor

a, a link of ex*[Esc]*

the vi full-screen editor, a link of ex

Insertion of Text at line extremes (I and A)

- **I** Inserts text at beginning of line
- **A** Appends text at end of line



Using **I** and **A** to Create a Comment Line in a C Program

/**********
****

To display above shown in vi editor

- Make use of repeat factor
- Step1: enter / in input mode
- Step2: Press [Esc]
- Step 3: then enter 20a*[Esc]

Opening a New Line(o and O)

o opens new line below current line

O opens a new line above current line

- Press [Esc] key after completing text input

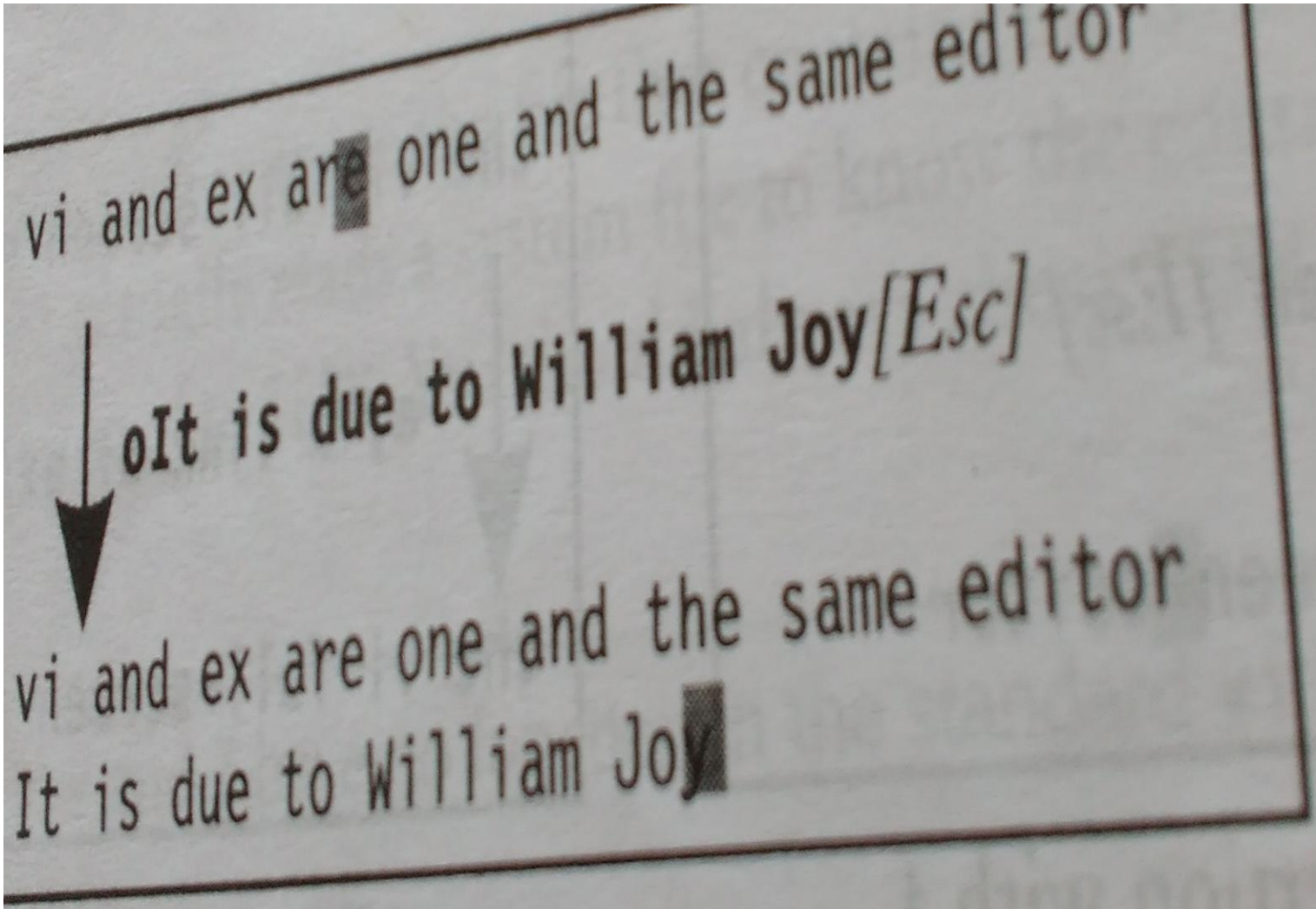


Fig. 7.6 Opening a New Line with **o**

Replacing text(r, s, R and S)

- **To replace a single character with another**, you should use

  r **No [Esc] required**

- Vi editor switches from command mode to input mode when r is pressed. It returns to command mode as soon as new character is entered. There is no need to press [esc]

- To replace letter a with one, make use key

  s Replaces one character with many

And press [Esc]

- To replace with multiple characters, use a **repeat factor**

  3s replaces three characters with new text

- R and S act on larger group of characters:

R Replaces all text on the right of cursor position

S Replaces the entire line irrespective of cursor position(Existing line disappears)

The ex mode-Saving text and quiting

- When we edit a file using vi or any editor—the original file isn't disturbed, but only a copy of it that is placed in a buffer(a temporary form of storage).
- From time to time , we should save our work by writing the buffer contents to disk to keep the disk file current
- Saving a file means saving this buffer.

Saving your work(:w)

- To save the buffer and remain in the editor
- :w command to write the buffer to disk

Note:

When you attempt to save the file with :w, vi displays with message File is read only.

Just save the file with a different name(:w fun)

After making sure that fun file doesnot exist.

Saving and Quitting(:x and :wq)

- To save and quit the editor(i.e return to shell), use the :x(exit) command
- Or use :wq to save and quit the editor
- The best way to save and quit the editor is to use **ZZ**

Aborting Editing(:q)

- To abort the editing process and quit the editing mode without saving the buffer

:q [Enter] //won't work if buffer is unsaved

No write since last change(:quit! Overrides)

:q! //Ignores all changes made and quits

- Vi suggests appending ! to an ex Mode command every time it feels that you could be doing something that is potentially unsafe.

Writing Selected Lines

- W can be prefixed by one or two addresses separated by a comma.
- The command

  :3,6w fun //saves lines 3 through 6 to the file fun
- To save a single line

  :1w fun1 // saves 1st line to file fun1

:. w tempfile //saves current line(where cursor is positioned)

:$w tempfile // saves last line

:., $ tempfile // saves current line through end

- If tempfile exists and is writabe by you, vi displays a warning

“tempfile” File exists – use “w! tempfile” to overwrite

! is universal overriding operator in ex mode

Escape to unix shell(:sh and [ctrl-z])

:sh

$_

Recover file from a Crash(:recover and –r)

- The functions of command mode:

  This is the mode you come to when you have finished entering or changing the text

  A command mode command doesn’t show up on screen but simply performs a function

Navigation

- Movement in four directions

k moves cursor up

j moves cursor down

h moves cursor left

l moves cursor right

- Repeat factor can be used as a command prefix with all these four commands

  4k moves the cursor 4 lines up

  20h takes it 20 characters to the left

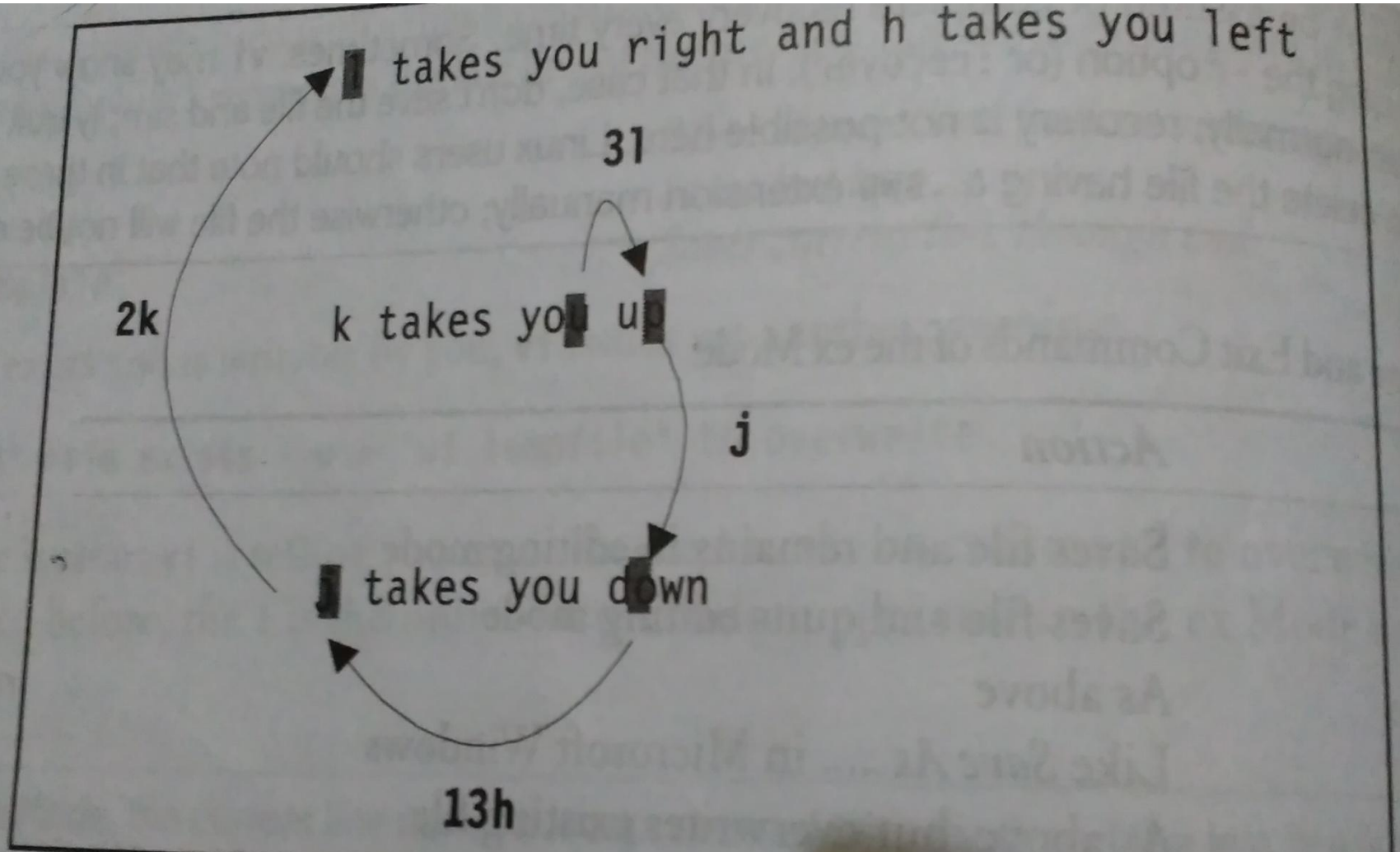


Fig. 7.10 Relative Navigation with **h**, **j**, **k** and **l**

Word navigation(b, e and w)

- **b moves back to beginning of word**
- **e moves forward to end of word**
- **w moves forward to beginning of word**

  repeat factor speeds up cursor movement along a line

For ex: **5b takes the cursor five words back**

3w takes the cursor three words forward

- Word is simply a string of alphanumeric characters and the _(underscore)

  sh_profile → one word

  sh-profile→ three words

  Keys B, E and W perform functions similar to lowercase (b, e , w) except that punctuation is skipped

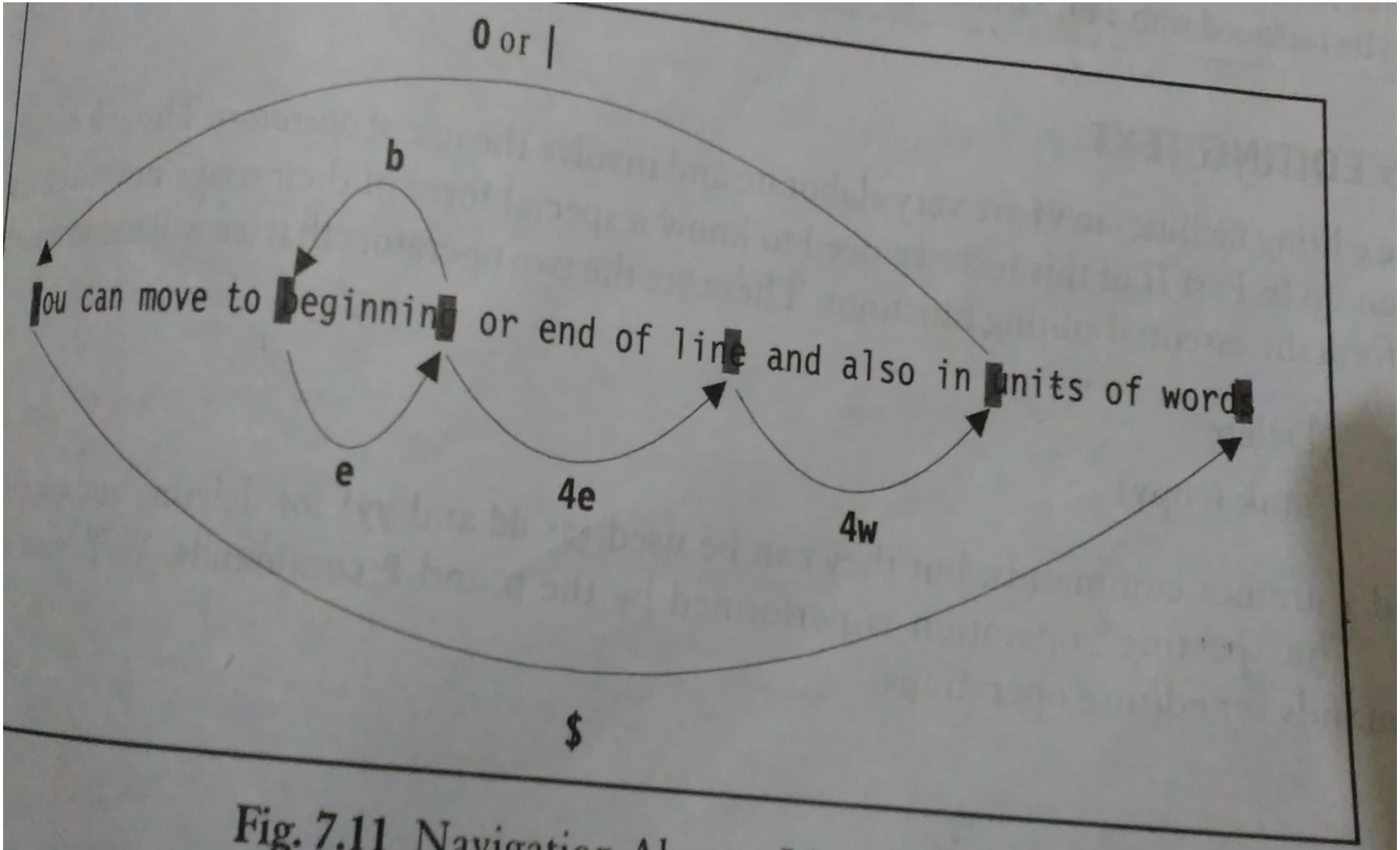


Fig. 7.11 Navigation Along a Line

Moving to line extremes(0, | and $)

- To move to the first character of a line

  0(zero) or |

The | takes repeat factor and it can

position the cursor on a certain column

30| //moves cursor to column 30

- $ in command mode indicates end of line

  $ //moves to end of line

Scrolling([Ctrl-f],[Ctrl-b],[Ctrl-d] and [Ctrl-u])

- Two commands for scrolling a page at a time

  [Ctrl-f] scrolls forward

  [Ctrl-b] scrolls backward

Use repeat factor,

10[Ctrl-f] -> scroll 10 pages forward

- To scroll by half a page

  [Ctrl-d] scrolls half page forward

  [Ctrl-u] scrolls half page backward

Repeat factor can be used

Absolute Movement(G)

- Vi editor displays the total number of lines
- At any time,you can press [ctrl-g] to know the current line number

"/etc/passwd" [Read only] line 89 of 179 ---49%

- Use of G command with repeat factor

  40G // goes to line number 40

  1G //goes to line number 1

  G //goes to end of file

Editing text

- Two operators: d delete

  y yank(copy)

- d and y are not commands, but they can be used (as **dd** and **yy**) for deleting and copying entire lines
- **p and P** commands used for pasting operation

Deleting text (x and dd)

- x command deletes the character under the cursor

  x //deletes a single character

  The character under cursor gets deleted, and the text on the right shifts left to fill up the space

  4x deletes the current character as well as three characters from the right

- Entire lines are removed with **dd** command

  6dd deletes the current line and five lines below

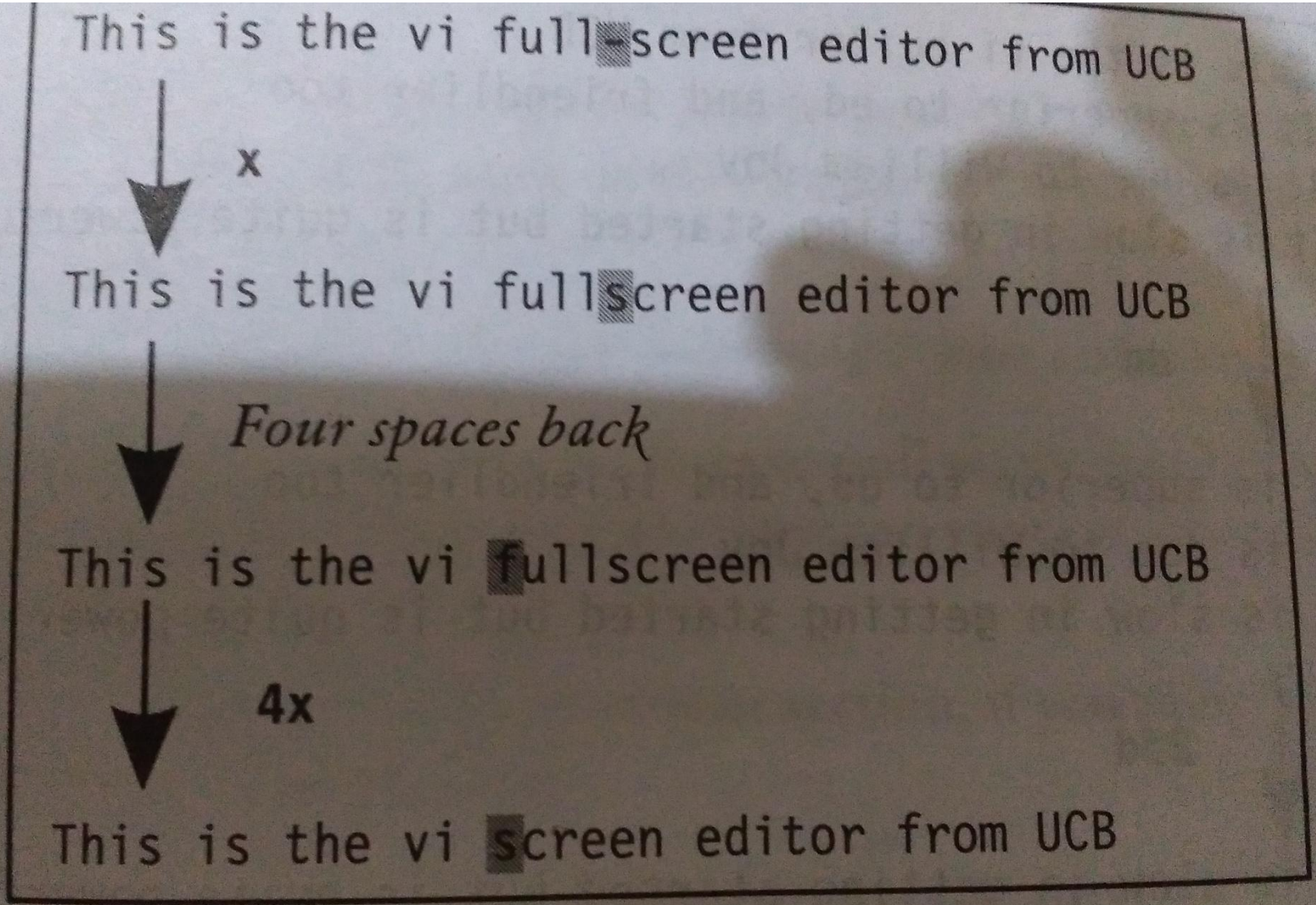


Fig. 7.12 Deleting Text with **x**

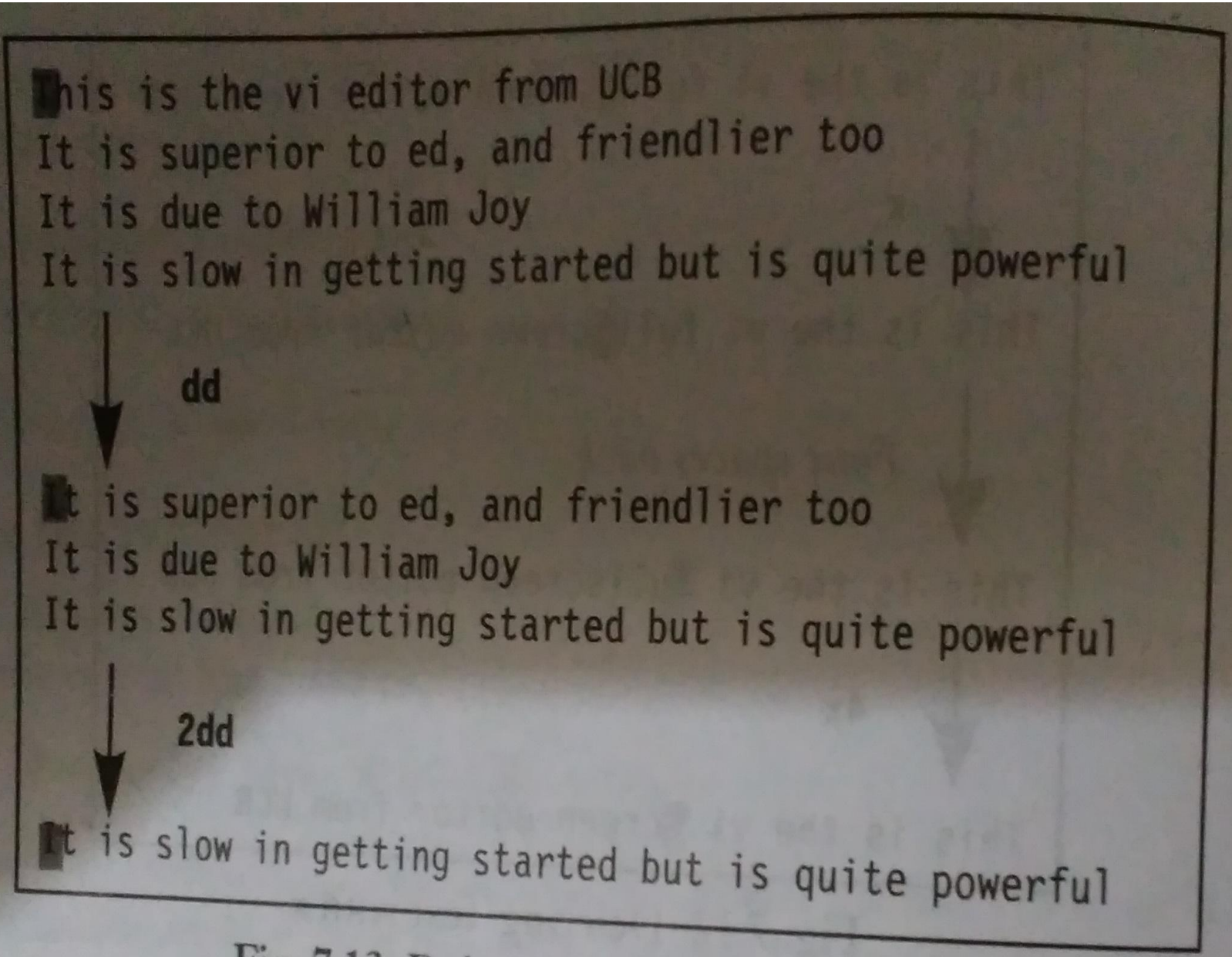


Fig. 7.13 Deleting Lines with **dd**

Moving text(p)

- The significance of p and P depends on whether they are used on parts of line or complete lines
- vi editor uses two commands for all “put” operations that follow delete or copy operations
- **p** places text below the current line and P places text above
- For example:

  x sdtio.h becomes stio.h – cursor on t

  p d put on right – stio.h becomes stdio.h

Copying text(y and p)

- Vi editor uses the term yanking for copying a text.

  yy //yanks current line

  10yy //yanks current line and 9 lines below

Joining lines(J)

- To join the current line and line following it,use **J**

  4J //joins following 3 lines with current one
- **J** removes the newline character between two lines to pull up the line below it

Undoing last editing instructions (u and U)

- Vi provides the u command to undo the last change made

  u //must use in command mode; press [Esc] if necessary

- When a number of editing changes have been made to single line, vi allows you to discard all changes before you move away from line

  U //don't move away from current line

- U reverses all changes made to the current line, i.e, all modifications that have been made since the cursor was moved to this line

Repeating the last command(.)

- .(dot) command is used for repeating both input and command mode command that perform editing task
- Ex :if you have deleted two lines of text with 2dd,then to repeat this operation,is to position the cursor at the desired location and press **u**

- Ex: You have to indent a group of lines by inserting a tab at the beginning of each line

  To use i[tab][Esc] only once, say on first line. You can then move to each line in turn by hitting [Enter] and simply press the dot.A group of lines can be indented in no time.

Searching for a pattern(/ and ?)

- vi editor is strong in search and replacement activities
- Searching can be made in both forward and reverse directions and can be repeated
- It is initiated from command mode by pressing a /, which shows up in the last line

  /pattern [Enter] //searches forward

- The search begins forward to position the cursor on the first instance of the word
- Vi searches the entire file, so if the pattern can’t be located until the end of file is reached, the search wraps around to resume from the beginning of the file
- If search still fails, vi responds with message Pattern not found

?pattern[Enter] //searches backward

- Searches backward for the most previous instance of the pattern.
- Wraparound feature also applies here but in the reverse manner

Repeating the last pattern search (n and N)

- For repeating a search in the direction the previous search was made with / or ?, use

  n //repeats search in same direction of original search

  N //repeats search in direction opposite to that along which previous search was made

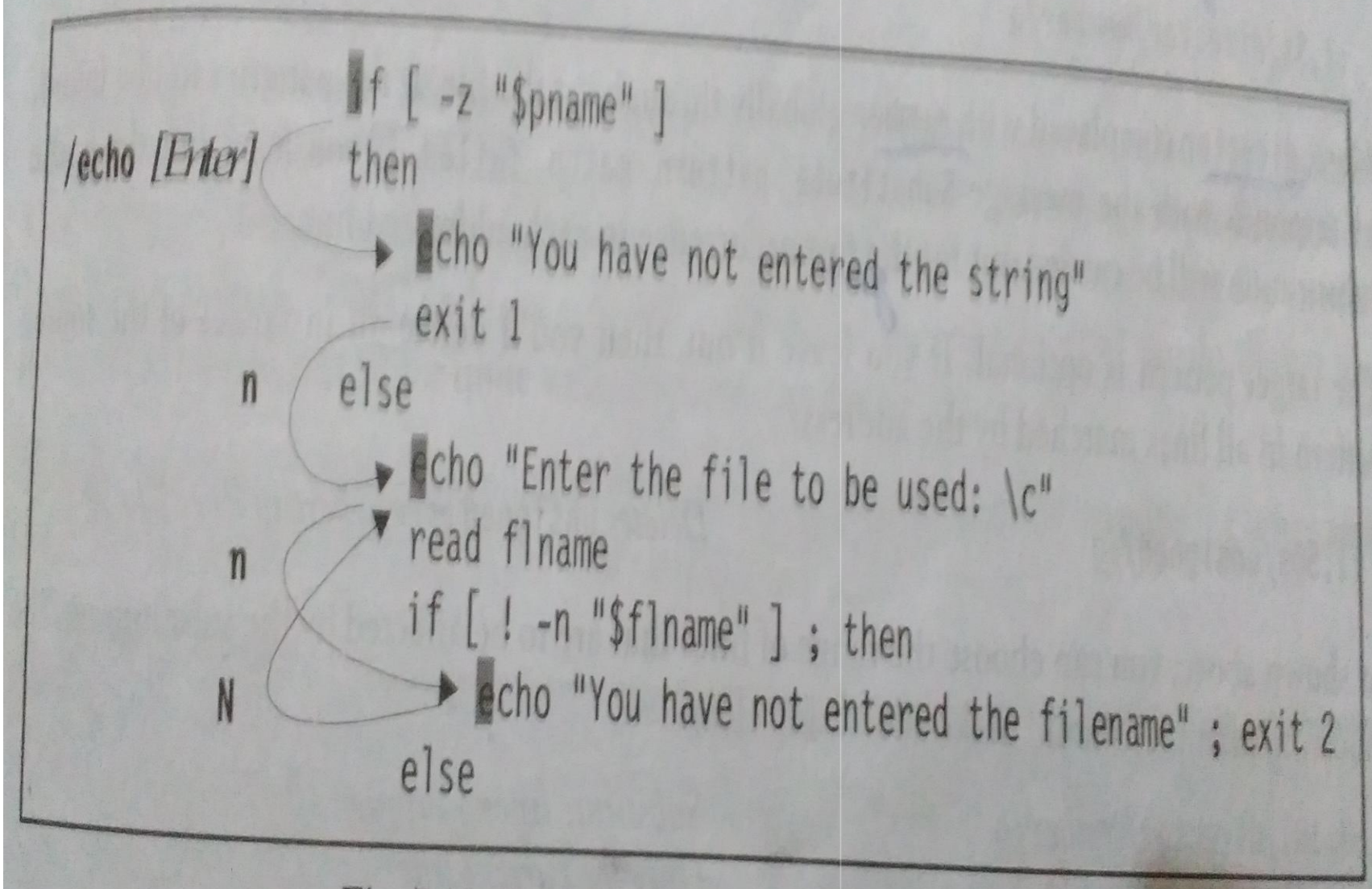


Fig. 7.14 Search and Repeat with / and n

Objective :To replace some occurrences of int with double

-> search for int with **/int** // search

-> repeat the search with n //repeat search

-> press the . //repeat last editing command

- **+/** symbols before pattern

  vi +/scheduling chap01 //cursor at scheduling
- The cursor will be located at the first instance of the pattern .You can then use n and N in usual way for locating the next instance of the string.
- If the pattern contains multiple words , enclose them with quotes

Substitution-search and replace(:s)

- Vi offers another powerful feature i.e substitution, which is achieved with ex mode s(substitute) command
- Substitution means replace a pattern in the file with something else.

- The syntax is

  :address/source_pattern/target_pattern/flags
- The source_pattern here is replaced with target_pattern in all lines specified by address
- The address can be one or a pair of numbers, separated by a comma.

  For example:1,$ addresses all lines in a file
- The most commonly used flag is g, which carries out the substitution for all occurrences of the pattern in a line

:1,$s/director/member/g

can use % instead of 1,$

director is replaced with member globally throughout the file