

The University of Texas at Austin  
Academic Computing and Instructional  
Technology Services

# Introduction to Vi

A self-paced tutorial for learning to use  
the vi screen editor

Short Course 602  
January 1998

## Introduction to Vi

ACITS  
The University of Texas at Austin

Copyright 1991-1998

Help Desk  
(512) 475-9400

The exercises on the following pages are for use in the ACITS “hands-on” short course to introduce the vi screen editor, which is available on the ACITS VMS Cluster (UTXVMS) and on ACITS UNIX systems.

NOTE: This handout assumes that you already know how to log in on the system you plan to use. If you do not, take the introductory short course for that system first. For information on obtaining documentation or signing up for short courses, call the ACITS help desk at 475-9400.

If you are enrolled in the short course, you will be provided with a temporary user number on the VMS Cluster or on UTS, depending on whether you plan to use OpenVMS or UNIX in the future. Slight differences exist between the versions of vi that are offered on these two systems.

### Before You Begin

1. If you are participating in the short course, the terminal you will be using for these exercises has already been set up as a VT100. For other types of terminals you can use on campus, follow the instructions appropriate to the terminal or workstation you are using.
2. The vi editor is case sensitive. Always use the same character case (e.g., lower case or upper case) shown in the command you are practicing.

NOTE to OpenVMS Users: As you probably know, DCL commands can be typed in either character case. (An exception is the terminal setting portion of a DEFINE TERM command, which sets the terminal type for UNIX-style software, such as vi.) Because this course is used on UNIX and OpenVMS systems, commands in this handout are shown in lowercase, which is required on UNIX.

Tasks that you should try out or practice are enclosed in boxes like this one.

- If you have trouble, ask the instructor supervising the short course for help.
- The footnotes in this handout are for those who are working without an instructor. If you are enrolled in the short course, ignore them.

§§ NOTE: Please work through the exercises in the order they are given. Skipping exercises  
§§ may cause incorrect settings in the next exercise. (In this document, the §§ marks at the  
§§ left indicate important text.)

Now you are ready to start working through the exercises.

## Contents

The topics discussed in this handout are organized so that you can use the commands you have learned in previous lessons to do the exercises in the current lesson. Thus, exercises should be done in the order presented. This list of the handout's contents is provided only for future reference. Don't use it to skip around among the topics.

Before You Begin .....	2
Contents .....	3
Typefaces and Symbols Used in This Document .....	4
A Practice File .....	5
Emergency Exit .....	5
Setting Up and Running Vi .....	6
Scrolling Backward and Forward .....	7
Simple Cursor Movements .....	8
Inserting Text .....	9
Deleting Text in Insert Mode .....	10
“Un-doing” and “Re-doing” Commands .....	11
Saving Your Work .....	12
Discarding Your Changes and Exiting from Vi .....	13
Suspending Vi .....	14
Resuming a Suspended Vi Session .....	15
Searching for Patterns .....	16
Deleting Text in Command Mode .....	17
More Advanced Cursor Movements .....	18
Substituting Text .....	19
Marks .....	20
Vi Command Syntax .....	21
Changing and Replacing Text .....	22
Copying and Moving Text .....	23
Moving or Copying Text Between Files .....	24
Mapping Commands .....	25
Other Useful Commands .....	26
Available Documentation .....	27
Finishing Up .....	28

## Typefaces and Symbols Used in This Document

When typing the commands shown in the exercises:

- Text shown in small boxes, such as `RETURN`, represents a key or key sequence you should type. Many vi commands are “control characters”. To type a control character:
  - Press and hold down the key marked CTRL.
  - Type the character (usually a letter, such as “f” or “b”).
  - Release the key marked CTRL.

An example of a control character is `CTRL f`, which is entered by pressing and holding the CTRL key while you type the letter “f”. Control characters execute immediately when you type them.

- Type the **boldface** words in command lines exactly as shown, including the character case.
- Substitute an appropriate value for items in *italics*. For example, substitute a UNIX filename or a OpenVMS file specification for *file* where it appears in a command line.
- Some vi commands execute immediately when you finish typing the command characters. If you must terminate the command by pressing the `RETURN` key, that will be indicated.

For example, a UNIX command (for copying a file from another directory into your own directory) and a similar OpenVMS command are shown below:

UNIX:                                `cp file . RETURN`

OpenVMS:                            `copy file [] RETURN`

---

§§ NOTE: The standard prompts for the operating systems on ACITS systems are

§§                            UNIX:        %

§§                            OpenVMS:    \$

§§ If your user number has a file named `.cshrc` (on UNIX) or `login.com` (on OpenVMS) in the top-level directory, this file may contain commands that change the prompt to something else. A common choice is the name of the system on which you are logged in.

For example, if you are using UNIX on UTS with the standard `.cshrc` file, you will see the prompt “uts.cc.utexas.edu> “. However, this is the same as “%” and anything that you can type at the % prompt, you can also type at the “uts.cc.utexas.edu>” prompt.

---

## A Practice File

Use the commands below to copy a file containing some text for you to edit.

### If you are using UNIX:

- Use the command below at the % prompt to copy your practice file. Give it any filename you want.

```
cp /usr/local/lib/shortcourse/602/vi.tasks file 
```

### If you are using OpenVMS:

- Use the command below at the \$ prompt to copy your practice file. Give the file any name you want.

```
copy sc:[602]vi.tasks file 
```

## Emergency Exit

(Use this if you get into trouble and need to bail out.)

§§ NOTE: The control characters used on OpenVMS and UNIX systems to escape from a  
§§ program do not perform those same functions from within vi. If you get into trouble and  
§§ you need to abort from the vi editing session, you can do so in almost any situation by  
§§ typing the vi command sequence described below.

```
 :q! 
```

Exits from insert mode, if necessary, and forces an exit from vi. You will be returned to the system prompt and none of your editing changes (since the last time you wrote out the file) will be saved. The key marked ESCAPE or ESC will probably be somewhere in the row of keys at the top of your keyboard.

## Setting Up and Running Vi

Before you run `vi`, you must set up your terminal correctly. This must be done once during each login session. To simplify matters, you can place the setup command in your login initialization file for the system (`.login` or `LOGIN.COM`). Once the setup commands are in effect, they do not need to be typed again during that login session (unless you deliberately change the settings) no matter how many times you run `vi`. Commands to set up and run `vi` are:

**setenv** **TERM** *termtype* `RETURN`

(On UNIX only, using the C shell—`csh`) Sets your terminal type to be *termtype*. Remember to use the same character case as is shown. Substitute a terminal type in lowercase for *termtype*.

**define term** “*termtype*” `RETURN`

(On VMS only) Sets your terminal type to be *termtype*. Use lowercase characters for *termtype*. The quotation marks are necessary.

**vi** *file* `RETURN`

(On both systems) Runs `vi` and edits the file named *file*. The file name or specification that you substitute for *file* must be appropriate for the system you are using.

### If you are using UNIX:

- Set up your terminal<sup>1</sup> by typing at the system prompt:

**setenv** **TERM** **vt100** `RETURN`

Remember to type the boldface characters exactly as shown, using lowercase for `setenv` and uppercase for `TERM`.

- Run `vi`, substituting the name of your practice file for *file* in the command below:

**vi** *file* `RETURN`

### If you are using OpenVMS:

- Set up your terminal<sup>1</sup> using the command below:

**define term** “**vt100**” `RETURN`

Be sure `vt100` is in lower case, and don't forget the quotation marks.

- To run `vi`, substitute the name of your practice file for *file* in the command below:

**vi** *file* `RETURN`

NOTE: Later on, if you forget to set up your terminal properly, `vi` will start in “open mode” and you will see only one line at a time. If this happens, exit from `vi`, be sure that your terminal is set up properly as shown above, and run `vi` again.

---

<sup>1</sup>If you are taking the course on your own terminal, you may need to omit this command or substitute a different terminal type for `vt100` in the `setenv` or `define` command. The default login initialization files for ACITS systems attempt to determine your correct terminal type. If you have trouble with some of the control characters, be sure your terminal type is defined correctly and that flow control is turned off (“NO XOFF”).

## Scrolling Backward and Forward

Now that you have run vi, you will see a window similar to the one shown below. A screenful of the text of the practice file is displayed in most of the window. The last line on the screen is an information line telling you the name of the file you are editing and the number of lines and characters in the file.

### Sample Terminal Screen

```
The text of your practice file
appears here in the window.  Because
the file is more than one screen
long, you cannot see all of the file
yet.  You will learn how to do that
in the exercise on this page.  At
the bottom of your screen is a line
showing the filename, the number of
lines, and the number of characters.
For example:
```

```
"vi.tasks" 49 lines, 2738 characters
```

`CTRL f`

Scrolls forward one screenful, so that you can read the rest of the file.

`CTRL b`

Scrolls backward one screen, so that you can see the text that was on the previous screen.

### On UNIX and OpenVMS:

- Read through the practice file. When you are ready to scroll forward, press `CTRL f` and notice that the information line disappears as you scroll.

An information line may reappear whenever vi or the system has something to tell you. The next time you scroll the screen, the line will vanish again.

- Use both `CTRL f` and `CTRL b` several times each. When you reach the end of the file, notice the tilde marks (~) that vi uses to represent non-existent lines.
- When you are accustomed to using these commands and are ready to continue, use `CTRL b` several times to return to the beginning of the file.

## Simple Cursor Movements

§§ NOTE: If you suddenly find that your characters are being inserted into the text instead of moving you around, you may have typed an **i** by accident and put vi into insert mode. §§ If this happens, press the `ESCAPE` key to return to command mode. Don't worry about §§ the text you inserted by accident.

The “cursor” is the small blinking line or box you see in the window. Most vi commands perform their function at the position of the cursor. Thus, you need to be able to move the cursor to the place in the file where you want to make changes.

The simplest cursor movement commands are:

- h** Moves the cursor one character to the left.
- j** Moves the cursor down one line.
- k** Moves the cursor up one line.
- l** Moves the cursor one character to the right.

The easiest way to repeat the command is simply to type the character several times. For example, to move down three lines, you could type:

**jjj**

You may remember from reading the practice file that you can precede a vi command with a number to repeat the command that number of times. That format is useful with cursor movement commands. For example, to move down 10 lines, you could type:

**jjjjjjjjj**

or simply:

**10j**

### On UNIX and OpenVMS:

- Practice using the **h**, **j**, **k**, and **l** commands to move around the screen.
- Practice repeating the characters to repeat the function.
- Practice using numbers in front of the commands to move around the screen.
- Combine the cursor movement command with the scrolling commands `CTRL f` and `CTRL b` and see how you now have access to the entire file.
- When you are ready to continue, use `CTRL f` several times to move to the end of the file so that you will be ready for the next exercise.

NOTE: On some (but not all) terminals, you can use the arrow keys to move your cursor also.

## Inserting Text

To insert text into a file using vi, you must go into “insert mode”. While inserting text, you need the following commands:

- i** Places you in insert mode at the current cursor position. As you type in your text, it will be inserted immediately to the left of the cursor. Any text to the right of the insertion point will be moved right as you type. When you reach the end of a line, press the `RETURN` key.

`ESCAPE`

Exits from insert mode and returns you to command mode.

NOTE: You must exit from insert mode before using the arrow keys and cursor movement keys. They will not work while you are in insert mode.

### On UNIX and OpenVMS:

- If the cursor is not already positioned at the end of your practice file, move there now.

You should see one or two lines of text and several lines containing only tilde marks (~). These marks are not actually present in the file. The vi editor uses these marks to indicate unused lines on the screen.

- Type **j** a time or two to move down toward the tilde marks until you hear a beep. You have now reached the end of your file.
- Use **k** to back up a line or two and position your cursor in the middle of a line of text.
- Press **i** to enter insert mode.
- Begin typing some text. Notice that it is entered at the left of your cursor. Don't worry if you make a mistake. You will learn how to delete mistakes on the next page.
- Notice that the existing text is moved right and long lines “wrap around” onto the next line. Press `RETURN` whenever you reach the end of a line. Avoid leaving wrap-around lines in your file. They can sometimes cause problems when you use your file with other programs.
- When you are ready to continue, type one last character “k” and notice that it is inserted as text. Typing “k” does not move your cursor up one line, as it would in command mode.
- Press `ESCAPE` to exit from insert mode.
- Type the character **k** again now. Notice that it now moves you up one line, because you are in command mode again.

NOTE: When using vi, avoid leaving spaces at the end of the line. These spaces can prevent some of vi's cursor movement commands from working properly.

## Deleting Text in Insert Mode

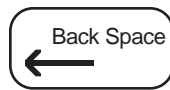
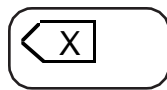
To correct errors that you make while typing in insert mode, use the `DELETE` or `BACKSPACE` key.

`DELETE` or `BACKSPACE`

(In insert mode) Deletes a character that you just inserted. You cannot delete back past the point where you entered insert mode. Characters are not removed from your screen immediately.

- If you insert more characters, the deleted characters will be overwritten.
- If you do not insert more characters, the deleted characters will disappear from the screen when you press `ESCAPE` to exit from insert mode.

The key you want is on your main keyboard (not on the keypads to the right of your main keyboard). The key may sometimes also be marked with a left-facing arrow in a box. Some examples of what this key may look like are shown below.



### On UNIX and OpenVMS:

- Press `i` to enter insert mode.
- Begin typing some text. Make several mistakes in a row.
- Use the `DELETE` or `BACKSPACE` key to delete the erroneous characters. Notice that the characters do not disappear as you delete them.
- When you have backed up over all the incorrect characters, begin typing again until you have overwritten all the errors with good text.
- Now type in an additional word that you can easily identify—your name, for example.
- Use the `DELETE` key to back up over the word, but do not type any additional text to obliterate it.
- Press the `ESCAPE` key. Notice that the additional word has been removed from the file.
- Experiment with insert mode and the `BACKSPACE` key until you think that you understand how it works.

## “Un-doing” and “Re-doing” Commands

The commands below can save you a lot of time. The “undo” command is particularly important when you make a text change and realize almost instantly that it was a big mistake. These commands are explained here because they may also be useful to you as you are working through this lab script. If you accidentally type a command incorrectly and delete a large portion of your practice file, you may be able to “undo” the command to reverse the damage and try again.

- u** Tells vi to “undo” the most recent text change you made.  
NOTE: The **u** command can undo only the most recent text change you made. Once you make another text change, **u** cannot affect any previous command.
- .** (Period) Tells vi to perform the last textual change again (“redo”) at the current cursor location.

NOTE: These two commands do not recognize cursor movement and scrolling commands. If you issue a “redo” or “undo” command after cursor movement or scrolling commands, the most recent textual change to the file will be reissued or reversed.

### On UNIX and OpenVMS:

Practice using the “undo” command:

- Press **i** and insert a word or two of text. Press `ESCAPE` when you are finished.
- Use **u** to undo the insertion.
- Use **u** again to undo the previous command (“undo”). Notice that the insertion is restored.

Practice using the “redo” command:

- Move to the beginning of a line and press **i** to enter insert mode again. Insert another word or two of text and press `RETURN`, so that your insertion is on a separate line. Press `ESCAPE` when you are finished.
- Move your cursor down several lines and press **.** (period) to redo the insertion at the current cursor position.
- Move to the bottom of the file and press **.** (period) three or four times in a row to insert the text several times.

## Saving Your Work and Exiting from Vi

The vi screen editor has an associated line editor named ex. Vi recognizes ex commands, which are preceded by a colon (:) to identify them. Many file manipulation tasks in vi are performed using ex commands. Thus, you will use ex commands to save your editing and exit from vi. If you make an error while typing an ex command, you can use the `DELETE` or `BACKSPACE` key to correct the mistake.

`:w file` `RETURN`

Writes your editing changes into a file named *file*. If you omit *file*, your changes are written to a new file with the same name as your old one. On UNIX, the old file is overwritten by the new one. On OpenVMS, a new version is created.

`:wq` `RETURN`

Writes your editing changes into a file by the same name as the one you started editing and exits from vi.

NOTE: **Do not** use `CTRL z` to exit from vi. This command creates another process instead of actually terminating the editor. You must then use the **attach** (OpenVMS) or **fg** (UNIX) command to return to the vi process. If you use `CTRL z` consistently to exit from vi, you will create many processes and wastefully consume the system resources that you and others need to do your work.

### On UNIX and OpenVMS:

- You have made a few minor changes to your file now, so exit from vi by typing:

`:wq` `RETURN`

- When you see the system prompt, run vi again by substituting the name of your practice file in the command:

`vi file` `RETURN`

§§ NOTE: If you get an error message after you try to save a file, you may have used up all  
§§ of your disk space. If this is the case, you may have to delete some old files before you  
§§ can save your editing. If this happens frequently, you may need to request more disk  
§§ space by sending mail to the mail address **remark**.

Later in this course, you will learn how to suspend vi, so that you can delete some old files to make room for the new one you want to save.

## Discarding Your Changes and Exiting from Vi

Sometimes you may find that you have done some editing in error and want to throw those changes away without writing out the file. This is particularly important on UNIX systems, where your new file may overwrite the existing file without making a backup.

**:q!** `RETURN`

Discards all editing changes (since the last time you issued a **:w** command) and exits from vi.

### On UNIX and OpenVMS:

- You should be at the beginning of your practice file. Type **i** to enter insert mode.
- Type the word “error” at the beginning of the file and press the `ESCAPE` key.
- Move to another location in the file and insert the word “error” again.
- Insert the word “error” at a third location.
- Pretend that the word “error” is some incorrect editing that you want to throw away. Notice that “undo” will not remove all three occurrences of “error”.
- Abort from your editing session, discarding your changes by typing:

**:q!** `RETURN`

- When you see the system prompt, run vi yet again by substituting the name of your practice file in the command:

**vi file** `RETURN`

- Look at your file. Notice that the occurrences of the word “error” were not saved in the file, but your previous changes (for example, the text you inserted in a previous exercise) are intact.

Note that the “emergency exit” command you were told about near the beginning of the script merely consists of pressing the `ESCAPE` key to get out of insert mode, followed by the **:q!** command.

## Suspending Vi

You may sometimes want to suspend vi for a moment while you check something or perform some administrative task. Unless you plan to return to vi almost immediately, write out your changes and exit from vi properly. Occasionally, however, you may need to exit temporarily—for example to delete some old files so that there is room to save your new editing.

### NOTE:

§§ If you suspend vi, **DO NOT** run vi again during the same login session. Instead, reconnect §§ to your vi process. Each time you run vi, a new process is created, and each process §§ consumes valuable resources needed to do your work and the work of others. Several §§ abandoned vi processes can cause a substantial decrease in response time for everyone.

`CTRL z`

Suspends the vi process and returns you to the system prompt. On UNIX, the vi process is stopped and you are returned to your original process. On OpenVMS, you are placed in a subprocess of your original process.

### On UNIX:

- Press `CTRL z` to suspend vi. You will see the system prompt (%).
- List your files by typing:

`ls -l` `RETURN`

**For future information:** On UNIX, you can delete old files using the UNIX command:

`rm file` `RETURN`

Use **man rm** on UNIX to learn more about this command.

### On OpenVMS:

- Press `CTRL z` to suspend vi. You will see the system prompt (\$).
- List your files by typing:

`directory` `RETURN`

- Note that each time you have written out the file, a new version was written. This feature of OpenVMS sometimes makes it necessary to suspend the editor and **purge** or **delete** files to make room for new versions.

**For future information:** On OpenVMS, you can delete old files using:

`delete filename.filetype;n` `RETURN`

`purge filename.filetype` `RETURN`

Use **help delete** (or **help purge**) on OpenVMS to learn more about these commands.

## Resuming a Suspended Vi Session

To resume a suspended vi process on UNIX, use the commands:

**jobs** Displays a list of your processes. Each will be preceded by a number.

**fg %n** Brings job number *n* to the “foreground”. You will be in the same position in your editing as you were when you suspended the process.

To resume a suspended vi process on OpenVMS, you must log out of the subprocess:

**logout**

Logs you out from the subprocess created when you suspended vi. You will be returned to your editing in the same place that you were before.

### On UNIX:

- To see a display of your processes, type:

**jobs** `RETURN`

- Find the one that says “Stopped” and mentions vi in its description. Note the number at the left of that job.
- To resume the suspended vi process, substitute that number (probably 1) for *n* in the command:

**fg %n** `RETURN`

- Notice that you are back in your vi file, at the same position you were before.

### On OpenVMS:

- To return to your suspended process, type the following command at the \$ prompt:

**logout**

- Notice that you are back in your vi file, at the same position you were before.

## Searching for Patterns

To search for a certain string or pattern of characters, use the commands below:

**/pattern** RETURN

Searches forward from the current cursor position for the named *pattern* and positions the cursor before it. If the pattern is not found in the forward direction, vi also searches from the beginning of file forward to the current cursor position. NOTE: *pattern* is case sensitive.

**?pattern** RETURN

Searches backward for the named *pattern* and positions the cursor at the beginning of the pattern. If the pattern is not found in the backward direction, vi also searches from the end of the file backward to the current cursor position. NOTE: *pattern* is case sensitive.

**n** Finds the next occurrence of the search pattern specified in the most recent / or ? command. The direction of search is taken from the previous search command (/ or ?).

**/** RETURN

Searches forward for the next occurrence of the search pattern specified in the most recent / or ? command.

**?** RETURN

Searches backward for the previous occurrence of the search pattern specified in the most recent / or ? command.

### On UNIX or OpenVMS:

- Search forward for the string “important” from wherever you are in the file using:

**/important** RETURN

- Use the **n** command to find the same word again and again from different locations in the file until you are comfortable with the search pattern that vi uses. Don’t press RETURN after the **n**.

- Now type:

**?brackets** RETURN

- Use the **n** command again. Notice that both the pattern and the direction of search have changed.

- Search for the next forward occurrence of “brackets” using:

**/** RETURN

NOTE: The *pattern* can include expressions that are not simply text strings, such as those below:

**^** Matches the beginning of a line. For example, **/^The** searches for the next occurrence of “The” at the beginning of a line.

**\$** matches an end of line. For example, **/no\$** matches “no” at the end of a line.

**\*** Matches any number of the preceding character (including none). For example, **/x \*b** matches any number of spaces (or no spaces) between an “x” and a “b”. The \* is most useful in doing substitutions, which you will learn about later. For example, you could search for strings of multiple spaces and reduce them to a single space.

**.** Matches any single character.

## Deleting Text in Command Mode

In insert mode, you can use the `DELETE` or `BACKSPACE` key to delete text that you have just inserted. In command mode, you can use `DELETE` or `BACKSPACE` to correct errors you have made while you are typing an ex command (for example, `:wq`) but not to delete text from your file.

To delete text from your file in command mode, use the following commands:

- x** Deletes the character under the cursor.
- dd** Deletes a line.
- D** Deletes from the current cursor position to the end of the line.

### On UNIX and OpenVMS:

- Use **x** to delete a few characters.
- Use **x** to delete another character and then use **u** to undo the deletion.
- Use **dd** to delete a line. Notice that even if your cursor is positioned in the middle of the line, the entire line (even to the left of the cursor) is deleted.
- Use **6dd** to delete six lines (the current line and the five following it).
- Move the cursor to the middle of a line.
- Use **D** to delete the rest of the current line. Notice that only text to the right of the cursor is deleted.

By now, you have made many changes to your file and it's probably pretty hard to read. To make the upcoming exercises easier, discard those changes before you proceed.

- Type:  
**:q!** `RETURN`
- When you see the system prompt (% or \$), run vi again using using **vi file** `RETURN` as before.

§§ NOTE: You already know enough commands to make almost any editing change in vi.  
§§ However, the more advanced commands on the following pages can help you do these  
§§ tasks faster.

## More Advanced Cursor Movements

Use the commands below to move more quickly to a different location in your file. Unless otherwise indicated, you can repeat these commands by preceding them with a number.

- w** Moves cursor to the beginning of the next word.
- b** Moves cursor to the beginning of the previous word.
- e** Moves the cursor forward to the end of the word.

NOTE: Normally, vi treats each punctuation character or each punctuation character followed by white space (any number of blank spaces or tabs) as a separator to define a word. If you type the command characters above in upper case (**W**, **B**, **E**), vi performs the same movements, but considers only white space to be a separator.

- 0** (Zero) Moves to the beginning of the current line. Do not precede this command by a number.
- \$** (Dollar sign) Moves to the end of the current line.
- ^** (Caret) Moves to the first non-whitespace character of the current line.
- G** Moves to the last line of the file.
- nG** Moves to the *n*th line of the file. For example, **1G** moves to the first line of the file.
- )** (Right parenthesis) Moves to the beginning of the next sentence.
- (** (Left Parenthesis) Moves to the beginning of the previous sentence.

NOTE: A period must be followed by two spaces or a `RETURN` to be recognized as the end of sentence.

- }** (Right brace) Moves to the beginning of the next paragraph.
- {** (Left brace) Moves to the beginning of the previous paragraph.

NOTE: A blank line indicates the end of a paragraph.

### On UNIX and OpenVMS:

- Try each of the cursor movement commands above. Use each command at least once.
- Precede some of the commands with numbers to see how they work then.

## Substituting Text

Often you may want to search for a pattern to substitute another piece of text for it. To do this in vi, use the ex substitution command:

```
:range/old-pattern/new-pattern/options RETURN
```

If you omit the *range*, vi operates only on the current line. Some useful range designators are:

.	The current line.
\$	The last line of the file.
1,.	From the first line of the file to the current line.
.,\$	From the current line to the last line in the file.
n1,n2	From line <i>n1</i> to line <i>n2</i> .
%	Every line in the file. Same as 1,\$

A pattern designation that is useful in this context is the caret (^), which represents the point before the first character in the line. Use the caret as the *old-pattern* to insert a piece of text at the beginning of a line.

Options are:

- g** (Global) Changes all occurrences of *old-pattern* that appear in each line. If this option is omitted, only the first occurrence on each affected line is changed.
- c** (Confirm) Asks about each occurrence and changes only the ones which you confirm with a “y” followed by a RETURN. To leave an occurrence unchanged, simply press the RETURN key.

### On UNIX and OpenVMS:

- Use the command below to search for all occurrences of “important” in the file and change each of those occurrences to “significant”:

```
:%s/important/significant/g RETURN
```

- Position the cursor near the middle of the file and change all the occurrences of “significant” after the the current cursor position back to “important”:

```
::,$s/significant/important/g RETURN
```

- Change all occurrences of the word “the” between line 6 and line 12 to “zzz”, using the **c** option. Be aware that without the **c** option, the command below would change even occurrences such as “these” to “zzzse”.

```
:6,12s/the/zzz/gc RETURN
```

The editor displays each occurrence, marking it from beneath with three carets (^^^). To change that occurrence, type **y** and press RETURN. If you find any occurrences of “the” used within a word, leave those unchanged.

- Use **1G** and then **}** to position the cursor after the first paragraph.
- Use the command below to insert “>” at the beginning of each line of that paragraph:

```
:1,s/^/>/ RETURN
```

## Marks

Sometimes you may want to mark a place in your text, so that you can come back to it quickly after you have performed some other editing task. You can have as many as 26 different marks, represented by the letters of the alphabet. NOTE: Marks do not last beyond the end of the editing session.

**ma**

Marks the current cursor position with the alphabetic character *a*. You can substitute any alphabetic character for *a*. You can mark different points in the file in one editing session by using different letters to name the marks.

**'a**

Positions your cursor at the mark designated by the alphabetic character *a*. (Note that the **'** character is a grave accent, NOT a single quote.)

**'a**

Positions your cursor at the beginning of the line containing the mark designated by the alphabetic character *a*. (Note that **'** is the single quote character.)

### IMPORTANT NOTE

If you are using Telnet on a Macintosh, the **'** character may not function correctly with the default settings. If you are enrolled in the short course, the instructor has probably already changed this setting for you. If not, to fix this, look in Telnet's **Edit** menu, select **Preferences**, then select **Globals**. Uncheck the box that says something like "Remap quote to ESCape".

### On UNIX or OpenVMS:

- Position your cursor in the middle of a line somewhere near the beginning of your file.
- Mark that position as "a" by typing:

**ma**

- Position your cursor near the end of your file (somewhere before the last paragraph).
- Mark that position as "z" by typing:

**mz**

- Now move back to mark **a** by typing a grave accent followed by an **a**:

**'a**

- Move to mark **z** by typing:

**'z**

These marks will remain in your file until you exit from the editing session, but they will not last between sessions. You will use these marks again later in the script.

- Move to the beginning of the line containing mark **a** by typing a single quote in front of the **a**:

**'a**

## Vi Command Syntax

NOTE: Read this page before trying any commands on pages 23-26.

Perhaps you remember reading in the practice file that many vi commands have the general structure:

*n***c***e*

where:

- n* (Number) Tells vi the number of times to repeat the command.
- c* (Command) Represents a vi command—usually a single character, such as **d** or **y**.
- e* (Entity) Represents a text entity. An entity can be a word, line, paragraph, mark, place in the file, etc.

You have already used the *number* portion of this command structure, and you know some commands. You actually already know some of the symbols for text entities. When you typed the cursor movement “commands” you learned earlier (e.g., **w**, **b**, **),**}, etc.), you were actually typing only the text entities. Because you omitted a vi command, the “default” command (to move the cursor past the indicated entity) was executed.

The most useful text entities are shown as the third (usually boldface) character in each syntax below. You will get a chance to use these entities in commands on the next pages. Certain commands that have a text entity implicit in their definitions, such as **x** (delete character), cannot accept another entity. The syntaxes below do not apply to those commands.

- ncw* Causes the vi command *c* to affect the next *n* words. For example, **3yw** “yanks” the next three words into a buffer. (You will learn more about the **y** command later in the course.)
- ncb* Causes the vi command *c* to affect the previous *n* words. For example, **4db** deletes the four previous words.
- nc* SPACEBAR Causes the vi command *c* to affect the next *n* characters. For example, **5d** SPACEBAR deletes the next five characters, starting with the character under the cursor.
- ncc* (Duplicating the command character) Causes the vi command *c* to affect the next *n* lines. For example, **6dd** deletes six lines and **4yy** yanks four lines into a buffer.
- nc)* Causes the vi command *c* to affect the next *n* sentences. If your cursor is in the middle of a sentence, only the portion to the right of the cursor is affected. For example, if your cursor is in the middle of a sentence, **2d)** deletes the remainder of the current sentence and the sentence following it.
- nc(* Causes the vi command *c* to affect the previous *n* sentences. If your cursor is in the middle of a sentence, only the portion to the left of the cursor is affected.
- nc}* Causes the vi command *c* to affect the next *n* paragraphs. If your cursor is in the middle of a paragraph, only the portion after the cursor is affected. For example, if your cursor is in the middle of a paragraph, **2d}** deletes the remainder of the current paragraph and the paragraph following it.
- nc{* Causes the vi command *c* to affect the previous *n* paragraphs. If your cursor is in the middle of a paragraph, only the portion before the cursor is affected.

## Changing and Replacing Text

NOTE: If you did not read the previous page, please do so now before continuing.

Although you can always change or replace text by using delete and insert commands to delete the old text and insert new text, it may be easier to do it all in one step. The commands below will do this.

**rx** (Replace character) Replaces the character under the cursor with the character *x*. You can precede this command with a number to replace more characters. Examples are:

**rw** Replaces the character under the cursor with the letter *w*.

**3rz** Replaces the character under the cursor and the two following characters with “*zzz*”.

**n***ce* Changes the text indicated by the next *n* occurrences of text entity *e*.

If *e* is measured in “sentences” or a shorter unit (e.g. words or characters) the end of the replacement field is indicated by a dollar sign (\$). Until you reach the dollar sign, the characters you type will overwrite the current characters. After the dollar sign, you will be in insert mode.

If *e* is measured in paragraphs, the text to replace is deleted immediately and you are placed in insert mode.

Examples are:

**cw** Changes the word under the cursor to the characters that you will enter. This is probably the most common form of the **c** command.

**5c** `SPACEBAR`  
Changes the character under the cursor and the four following characters to the text that you will enter.

**R** (Replace mode) Places you in replace mode, so that each character you type overwrites the character under the cursor. Press `ESCAPE` to return to command mode.

### On UNIX and OpenVMS:

- Move your cursor to the beginning of the next line and use **15r\*** to replace the first 15 characters of the line with asterisks.
- Move your cursor to the beginning of another line and type **3cw** to change the next three words. Note the dollar sign (\$) that marks the end of the text you will be replacing.
- Begin typing some replacement text. Go slowly, and watch what happens when you reach the dollar sign. The dollar sign will be replaced, but thereafter you will be in insert mode and no further characters will be overwritten.
- Continue typing until you reach the end of the line. Press `RETURN` and continue typing in insert mode for at least a few more characters.
- Press `ESCAPE` to return to command mode. Move the cursor to the middle of a paragraph.
- Use **c}** to change the text of the rest of the paragraph. Notice that the text is immediately deleted. Type some text and notice that you are now in insert mode.
- Press `ESCAPE` to return to command mode.
- Move to the start of the last paragraph and type **mx** to establish a mark named **x**. Position the cursor at the beginning of the next-to-last paragraph.
- Change the text between the cursor position and the mark **x** by typing **c’x**. Type in the text you want to replace the paragraph with. When you finish, press the `ESCAPE` key.

## Copying and Moving Text

To copy text from one location to another, you must “yank” it from its current position and “put” it in the new position. To move text, simply use the **d** command to remove the text and “put” it in the new location.

- n $y$ e** (Yank) Copies the next  $n$  text entities of type  $e$  from the current location and places that text in a buffer (a text holding area). For example, **3yw** yanks the next three words, starting with the word immediately under the cursor. **3yy** yanks the next three lines.
- n $d$ e** (Delete) Deletes the next  $n$  text entities of type  $e$  from the current location and places that text in a buffer. You already know this command from using **6dd** before to delete the next six lines.

Remember that doubling the command character (**dd** or **yy**) causes vi to use the text entity “lines”.

- p** (Put) Places the most recently yanked or deleted text into the file *after* the current cursor location.
- P** Places the most recently yanked or deleted text into the file *before* the current cursor location.

### On UNIX and OpenVMS:

The yank, delete, and put commands are often used with marks, which you have already learned to use. Remember the marks “a” and “z” that you established earlier? Unless you have logged out for some reason in between that exercise and this one, the marks are still in your file. If they have been lost, you can re-create them, or simply use any two different locations.

- Go to mark “z” (near the end of your file) using the command:

**‘z**

- Use **3yw** to yank the next three words. You will see no visible change as the words are put into the buffer.
- Go to mark “a” and put the yanked text at that location by typing:

**‘aP**

Notice that this is simply the command **‘a** followed by the command **P**.

- Use **‘z** to return to mark “z” and **3yy** to yank the next three lines.
- Use **1G** to go to the beginning of the file and **P** to insert the yanked lines there.
- Use **}** to move to the beginning of the next paragraph and use **d}** to delete the paragraph.
- Use **‘z** to move to mark “z” and use **p** to insert the deleted sentences at that location.
- Position your cursor in the middle of a sentence and use **2d** to delete the next two sentences. Notice that for the first sentence only the portion of the sentence *after* the cursor position was deleted.

NOTE: Avoid using “backwards” entities (**b**, for example), when using marks to move text. Since your cursor is positioned immediately after the mark, you can yank or delete your mark at the delete point, and overwrite the mark at the target point when you “put” the text at the new location.

## Moving or Copying Text Between Files

Use the commands below to move or copy text from one file to another:

- `:r file` RETURN  
(Read) Inserts the text of the file named *file* into the file you are editing (after the current line.)
- `:e file` RETURN  
(Edit) Brings in a new file for editing. If you have unsaved changes in your current file, vi will warn you so that you can save them before you exit. If you choose to override the reminder, your unsaved changes to the current file will be lost.
- `“a o` Places the text from an operation *o* into a buffer (text holding area) named *a*. Use any alphabetic character for *a*. The operation *o* can be any vi command that defines an area of deleted, changed, or yanked text. For example, `“z6dd` deletes six lines and places them into a buffer named *z*.
- `“ap` “Puts” the text from buffer *a* immediately after the current cursor position.

NOTE: Text buffers are saved until you exit from vi. Thus, you can delete or yank text from one file, place it in any of 26 named buffers (a-z), and bring in another file to edit. All the text buffers will still be in place, and you can insert that text into the new file as needed.

- Use `:e` to bring in a new file by substituting a new filename for *file* in the command:  
`:e file` RETURN
- You should be warned that you have unsaved changes. Save the changes to your current file by typing:  
`:w` RETURN
- Now use `:e` again to bring in the new file. This time, vi should bring in the new file without protest. Because this is a new file, you will see a screenful of tilde marks (~).
- Type `i` and then insert some text. Press the ESCAPE key.
- Save the changes by typing:  
`:w` RETURN
- Use `:e` to bring in your original practice file again. Notice the position of the cursor and the text around it.
- Import the text of the new file you created to the current cursor position by typing:  
`:r file` RETURN

## Mapping Commands

The **map** command allows you to save time by mapping multiple vi commands to a single character or key sequence. The syntax is:

```
:map single-character vi-command-string RETURN  
Maps the single character to execute the indicated string of vi commands.
```

Example:

```
:map CTRL v CTRL w dw RETURN  
Maps the command CTRL w to the dw command (deletes a word).
```

NOTE: The **CTRL v** is a “quote” character that causes the control or escape character immediately following to be accepted as text instead of executed.

- Type the command below:

```
:map t j0wwdw RETURN
```

The command **t** now represents the command “Jump to the next line (**j**) at the beginning (**0**), skip two words (**ww**), and delete one word (**dw**)”. The character **t** is frequently used because most other characters already have assigned functions.

- Now position your cursor at the first word of a paragraph. Type **t** and confirm the action of the newly mapped command.
- Type the command below, which illustrates how to use the **map** command to insert text:

```
:map CTRL v CTRL w 0itext CTRL v ESC j RETURN
```

The command **CTRL w** now represents the command: “Return to the beginning of the line (**0**), enter insert mode (**i**), insert the word ‘text’, exit from insert mode ( **CTRL v** **ESC** ), and jump to the next line (**j**)”.

- Try out this new **CTRL w** command.

## Other Useful Commands

<code>CTRL l</code>	(Letter “l”) Refreshes the screen display.
<code>CTRL d</code>	Scrolls forward one-half screen.
<code>CTRL u</code>	Scrolls backward one-half screen.
<code>CTRL e</code>	Scrolls forward one line.
<code>CTRL y</code>	(UNIX only) Scrolls backward one line. DO NOT use this in vi on a OpenVMS system. Doing so can cause all kinds of problems.
<b>H</b>	(High) Moves cursor to first line of screen.
<b>M</b>	(Middle) Moves cursor to middle of screen.
<b>L</b>	(Low) Moves cursor to last line of screen.
<b>a</b>	(Append) Begins inserting text after the cursor position. To exit from insert mode, press the <code>ESCAPE</code> key.
<b>I</b>	Begins inserting before the first non-space character of the current line.
<b>A</b>	Begins inserting at the end of the current line.
<b>o</b>	Opens a new line under the current line, placing you in insert mode.
<b>O</b>	Opens a new line above the current line, placing you in insert mode.
<b>U</b>	Tells vi to “undo” all changes to the current line. Once you move the cursor off the current line, the <b>U</b> command cannot undo the changes.
<b>%</b>	Searches (in either direction) for the parenthesis, bracket, or brace that matches the one under the cursor. If the character under the cursor is not one of the paired punctuation marks, you will hear a beep and the cursor will not move.
<b>J</b>	(Join) Attaches the beginning of the next line to the end of the current one, inserting a space between them. Any leading white spaces are removed from the second line. However, two spaces will be left after a period.
<code>CTRL g</code>	Redisplays the line at the bottom of the screen that provides information about your file.

- Move your cursor around in the file and press `CTRL g` each time. Notice that the line number shown in the information line tells where you are in the file.
- If you have time, try out some of the other commands shown above.

NOTE: (On UNIX only) If you are disconnected while editing a file, you may be able to recover some editing. An automated mail message sent from the system will provide instructions on recovering the file. This does not guarantee that the file can actually be recovered.

Log in on the same system that you were using when you lost your editing and type **vi -r** to see if there are any recoverable files. To recover the editing on one of the filenames displayed, substitute it for *file* in the command:

**vi -r file**

## Available Documentation

### Printed Documentation:

Pocket Reference Lists are available at no cost from the Help Desk and from most ACITS output sites:

*Ex/vi Quick Reference List*  
*UNIX Pocket Reference List (CCRL-20)*  
*VMS Pocket Reference List (CCRL-21)*

Introductory manuals for UNIX and OpenVMS are available for purchase for approximately \$2.50 at the Help Desk.

*Introduction to UNIX (CCUG-1)*  
*VMS User Guide (CCUG-2)*

### Online Documentation:

On UNIX systems:

<b>man vi</b>	vi man page description
<b>man ex</b>	ex man page description

On OpenVMS systems (use **type** or **print/queue=queuename** to display or print SYS\$DOC documents; use **uthelp sites** to obtain a list of available queue names):

<b>help vi</b>	Setup information for ex and vi.
<b>sys\$doc:exrm.doc</b>	Ex Reference Manual (approximately 34 pages)
<b>sys\$doc:viin.doc</b>	An Introduction to Display Editing with Vi (approximately 40 pages).
<b>sys\$doc:vicommands.doc</b>	Vi Command Reference (approximately 38 pages)

## Finishing Up

You have now finished the scheduled exercises for the short course. If you have additional time, please do the following:

- Examine the ex/vi Pocket Reference List you were given at the start of class.
- Find the commands that you already know, so that you can locate them quickly if you forget the command syntax.
- Experiment with some of the commands you have not yet tried.
- If you have any questions about vi, ask the short course instructor<sup>2</sup>.

### NOTE:

**When you are finished with the short course, be sure to exit from vi, log out from the system, and close your network connections.**

### On UNIX and OpenVMS:

- Exit from vi by typing the vi command:  
**:wq**
- Be sure to log out from the UNIX or OpenVMS system when you are finished.
- Close any network connections you may be using (CCNET, Telnet, etc.) with the appropriate “Quit” or “Exit” command.
- If necessary, log out from the workstation you are using (for example, one of the workstations in the Student Microcomputer Facility).

**When you have finished, please complete the evaluation form at the end of the handout and give it to the instructor or mail it to ACITS through campus mail or U.S. Mail.**

---

<sup>2</sup>If you are taking the course on your own and you have questions, send electronic mail to **remark**. You will get a response within a day or two. If you need immediate assistance (for example, because you cannot log out), call 475-9400.

## Short Course Evaluation Form

### ACITS Short Course 602: Introduction to Vi

- What is your status?  Graduate Student  Undergraduate Student  
 Faculty  Staff  Non-UT
- Where did you learn about this course?  
 Daily Texan  Graduate Orientation  ACITS Help Desk  
 Faculty Adviser  Colleague  Other (please specify below)

---

- Date you took the course: \_\_\_\_\_ • Date on your copy of the script: \_\_\_\_\_
- Would you recommend this course to others?  Yes  No
- Did you take the course:  With an instructor present?  On your own?

#### If an instructor was present:

- How would you rate the instructor's assistance?  
 Excellent  Good  Satisfactory  
 Fair  Poor  Not required
- Additional comments on instruction: \_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

#### If you worked on your own:

- Were you able to complete the course to your satisfaction?  Yes  No
  - Did you feel that you needed an instructor?  
 Yes  No
  - If yes, when?  
 Throughout the course  
 At the point described below:  
\_\_\_\_\_  
\_\_\_\_\_
- Continue on back of form (interior fold), if needed.

#### Please rate the course on the points below:

- |  | Excellent                | Good                     | Satisfactory             | Fair                     | Poor                     |
|--|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| • The overall quality of this short course was:  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| • How well did the course contents meet your needs?  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| • The presentation of material in the script was:  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| • The print quality of the script you received was:  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| • How would you rate the general degree of difficulty of the script materials?<br><input type="checkbox"/> Much too difficult <input type="checkbox"/> Difficult <input type="checkbox"/> About right <input type="checkbox"/> Easy <input type="checkbox"/> Much too easy |                          |                          |                          |                          |                          |
| • If any part of the script gave you trouble, please describe the problem: _____<br>_____  |                          |                          |                          |                          |                          |
| • If any parts of the script were particularly useful, please list them here: _____<br>_____   |                          |                          |                          |                          |                          |
| • Can you suggest other short courses that you would like to see offered? _____<br>_____   |                          |                          |                          |                          |                          |
| • Other suggestions or comments: _____<br>_____  |                          |                          |                          |                          |                          |

Please return this form to ACITS through campus mail or U.S. Mail. Be sure to fold it so that the appropriate address shows on the outside. Don't forget to put a stamp on forms sent by U.S. Mail.

Short Course Administration  
ACITS  
G2700

CAMPUS MAIL



Place  
Stamp  
Here

Short Course Administration  
ACITS (G2700)  
The University of Texas at Austin  
Austin, TX 78712-1110