the CP/M handbook with mp/m

rodnay zaks

SYBEX

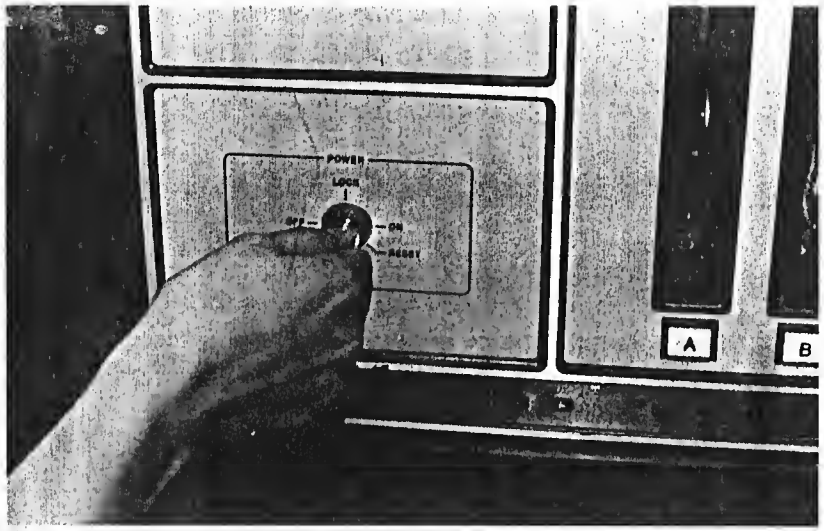


Figure 1.9: Turning Cromemco On

Figure 1.10: The Keyboard of the Terminal

it two or three times (see Figure 1.10). Suddenly, the system message and a *prompt* will appear on your screen:

System Message: 48K CP/M
System prompt: A.

or (with MP/M)

System Message: xxK MP/M
System prompt: 0A.

CP/M is now up and running and waiting for your commands.

Turn on the SOL

To turn on the SOL computer (an older system) use the switch on the back of the terminal, and turn on the TV monitor (CRT screen). (See Figure 1.11.) Turn on the separate disk drives and insert the CP/M System Diskette into drive A (the lower slot). (See Figure 1.12.) This symbol will immediately appear on your screen:

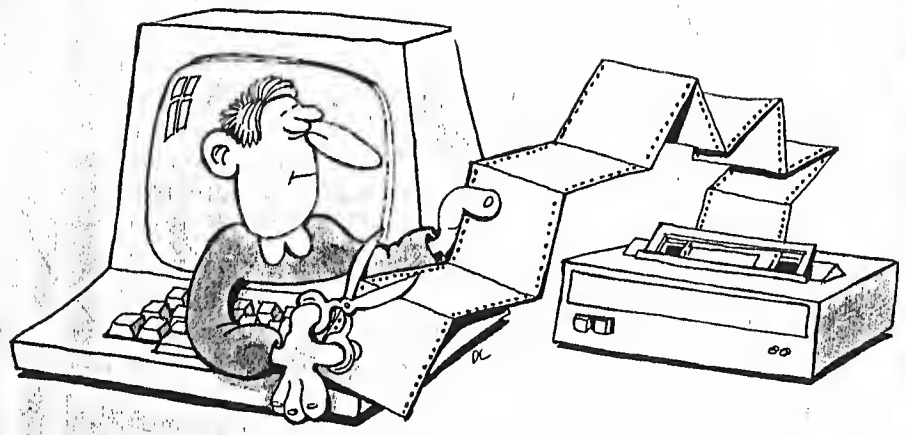
>

This is a prompt from the SOL's monitor program, not from CP/M, which is still on the diskette. Note that if the key marked LOCAL (on the SOL's keyboard) is ON, you are not actually connected to the system. Turn LOCAL off by pressing it.

To bring up the system, type the following command:

> EX E000 ↵

The symbol ↵ represents the RETURN key. The value E000 is the address of the program that loads CP/M automatically from the disk. This value varies with each disk controller. The vendor of your disk controller will tell you which address must be used with its system.



4

USING THE EDITOR

INTRODUCTION

Chapters 1 and 2 were designed to teach you everything you need to know to begin using CP/M. Chapter 3 described the most important utility program, PIP. This chapter will teach you how to use another important application program that comes with the CP/M operating system: the editor, ED. You will be shown how to use an actual editor program, and you will follow the data transfers between the disk, the computer's memory, and the terminal.

It is not important to remember the specific commands provided by ED. These commands are summarized in the Appendix section. What is important, however, is for you to understand how an editor operates, and what it can do. If you achieve this goal, you will find most other application programs simple to understand (if they are well-designed and documented). Also, you will probably be able to easily use a word processing program, a very useful application program on any computer.

This chapter is useful, but not indispensable. If you feel you are not interested in learning about the editor, you can go on to the next chapter.

WHAT IS AN EDITOR PROGRAM?

A good editor program allows you to create and edit text files — letters, novels, poems, business forms, or anything comprised of characters. The program should also let you move from line to line easily, and change characters by retyping over them, or by deleting and inserting characters in one motion. It should also be able to find any group of characters you specify in a file, and do substitutions. A good editor program should allow you to merge two files as well as interweave lines of text from two files.

When you type text using a good editor program, you will type a line and end it with a Carriage Return (RETURN or CR on some key-

boards), just as you would on an electric typewriter. In the future, an editor program might even make it easier than that; we have certainly not yet seen the best editor program nor the easiest one to use.

ED, the editor program provided with CP/M, is only a minimal editor and is not as easy to use as most other editor programs. If you plan to do a significant amount of editing or word processing, you should obtain a more powerful editor. There are many editor and "word processing" programs on the market today that will run on CP/M or MP/M.

A *word processor* is a program that includes both an editor program (for typing text) and a program that runs the printer (for printing text), making it backspace, underline, justify margins, expand tabs, and type in boldface type. Take note, however, of exactly what you are buying. There are some so-called "word processors" that are only printing programs designed to be used with CP/M's ED program. These are only formatters, and are not convenient or powerful enough for general use as word processors.

Shop around as if you were buying a typewriter. You might want bells and whistles; or, you might want a portable, inexpensive model that requires more effort but will do the job. However, ease of use should be your primary consideration (especially if you use one for writing). After reading this chapter, you will know what the minimum set of facilities provided by an editor should be. ED is sufficient for most simple applications.

ED, THE EDITOR

The editor program ED.COM usually resides on the System Diskette, and is executed by typing 'ED', followed by the name of the text file that you are creating or modifying. For example, if you want to create or modify the file SAMPLE.TXT, type:

```
A>ED SAMPLE.TXT↵
```

Do not type just 'ED'. This is a common error, and will not work. You must supply a file name.

NOTE: if you get the message 'FILE IS READ/ONLY', or 'SYSTEM FILE NOT ACCESSIBLE', then you have to use the STAT command on the file first (CP/M version 2.2 and MP/M). See the section on CP/M version 2.2 and MP/M in Chapter 2.

If ED cannot find the file that you specify, it assumes that you are

creating a new file. This file, specified in your command, becomes the "source" file. Subsequent ED commands will then bring the source file's text into the *edit buffer,* as shown in Figure 4.1.

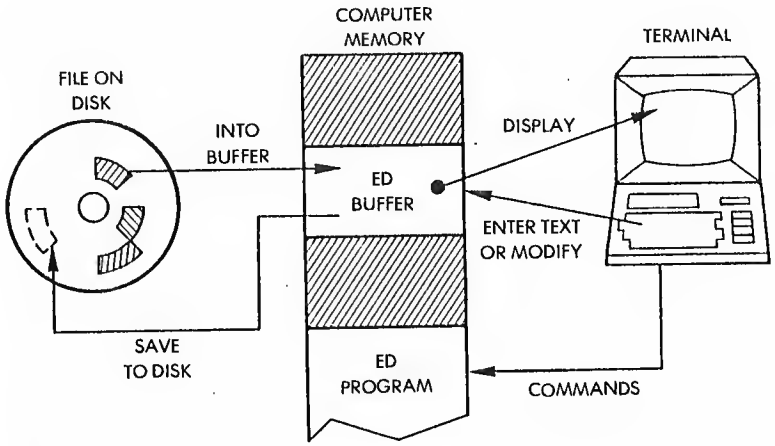


Figure 4.1: ED's Buffer

The *buffer* is a block of memory inside the computer reserved for ED's text processing. If your text file is large, you can load only one block of it at a time into the buffer. (Note that this is an inconvenience inherent to ED that does not exist in more powerful editors.)

You can only type new text in the edit buffer, or change text that is already in it. However, the buffer is not copied back to the disk automatically. If you terminate ED (or if you turn off the system) without *saving* the text that is in the buffer on a disk file, you will lose the text in the buffer. Since ED *copies* the text file into the edit buffer, your original file (i.e., before it was used with ED) is untouched but your new text and modifications are lost. Therefore, you should periodically save the text in the edit buffer, and always remember to save the buffer before leaving the ED program. You save the buffer by using ED's 'E' command (E↵). The E command also copies the rest of the source file into the new "source" file that it creates. (This is explained in more detail later in this chapter.)

Most of ED's commands consist of a special letter preceded by a number or symbol determining an amount. These commands are ex-

ecuted by typing them as you would type CP/M commands: the command, followed by a RETURN (↵), which transmits the command. Other "commands" are special key combinations (like CTRL and Z (↑Z), CTRL and C (↑C), etc.) that are transmitted automatically and do not require a RETURN.

The ED program continually displays the ED prompt:

```
*
```

The E command would be typed like this:

```
*E↵
```

Several commands act on the text in the edit buffer, while others transfer text to and from the edit buffer. The edit buffer is illustrated in Figure 4.2:

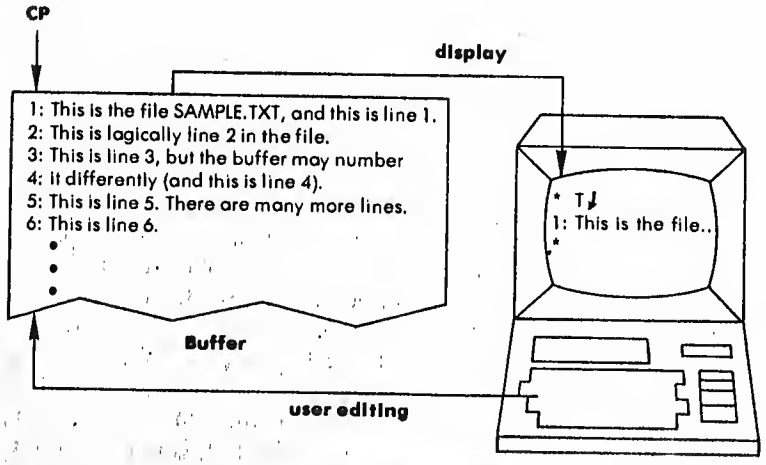


Figure 4.2: Text Being Processed

THE 'CP' (CHARACTER POINTER) AND LINE NUMBERS

The 'CP' in the illustration is the "character pointer." It is not actually displayed on the screen, but is used by the ED program, and is

moved by ED commands. The pointer always points at one character, and the ED commands usually refer to the characters following (and including) the character pointed to by the CP, or the characters trailing the CP. In other words, you move the CP to the *right* to go forward in the text, and you move the CP to the *left* to go backwards. You will see examples illustrated in subsequent sections of this chapter.

In addition to the imaginary CP, each line has an imaginary *line number* that is not actually part of the text. If you have the newer version of ED, it automatically displays the line numbers with the text (and you can move to any line of text by specifying the line number as a command, as shown later). If you have the older versions of ED, you have to *turn on* the display of line numbers by executing the V command (the command -V (negative V) turns it off). If you have even *older* versions, you might not have line numbers at all (which is unfortunate, since they do make it easier to move around in the edit buffer). The line numbers, like the CP, only exist in the edit buffer and are used only to move around in the buffer. They would not appear in a print-out of the file.

WHAT ED DOES TO YOUR TEXT FILE

For example, assume that the file SAMPLE.TXT already exists, and contains the text shown in Figure 4.3:

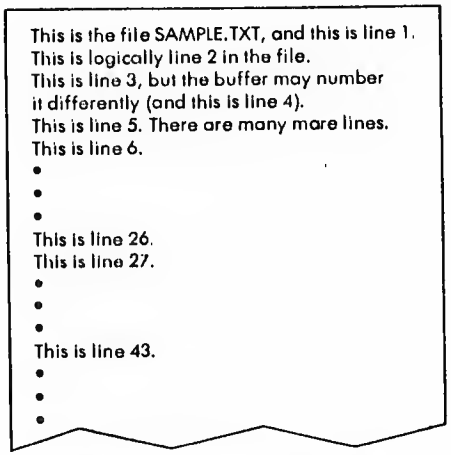


Figure 4.3: A Sample File Is In the Buffer

When you execute the CP/M command:

A> ED SAMPLE.TXT ↵

ED sets aside a space in the *transient program area* (scratch-pad memory—this will be explained in Chapter 5) for the edit buffer, creates a temporary output file called SAMPLE.$$$ (to store the edited file without damaging the original one), and prepares SAMPLE.TXT to be copied into the edit buffer. The copying is actually done by using the A (append) command—this will "append" a certain number of lines into the edit buffer, as shown in Figure 4.4:

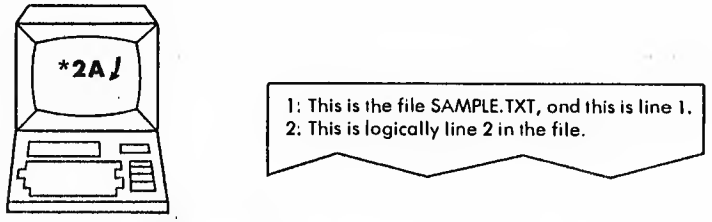


Figure 4.4: The Append Command is in A

The ED command '2A' appended the first two lines of the text file into the buffer. If you want to append more lines, you have to do another A command. This is shown in Figure 4.5.

Now you can modify these lines that are in your edit buffer. You can also add new lines from the keyboard (by using the I command, described later on in this chapter). This is shown in Figure 4.6.

You have now learned how to place lines into the buffer from the disk file or the keyboard. This is shown in Figure 4.7.

You could have appended the entire file into the edit buffer (unless the file has more than 6000 characters, and you are using a 16K version of CP/M), but for the sake of simplicity, this example sends the newly-modified text (with your new additions) to a *temporary output* file, so that you can append more text to the edit buffer. ED calls this temporary output file SAMPLE.$$$. To send the newly-modified text to the temporary output file, you use the W (write) command, or you perform a "normal" termination with the E (end edit) command or the H (end and reopen) command. Figure 4.8 shows the use of the W command.

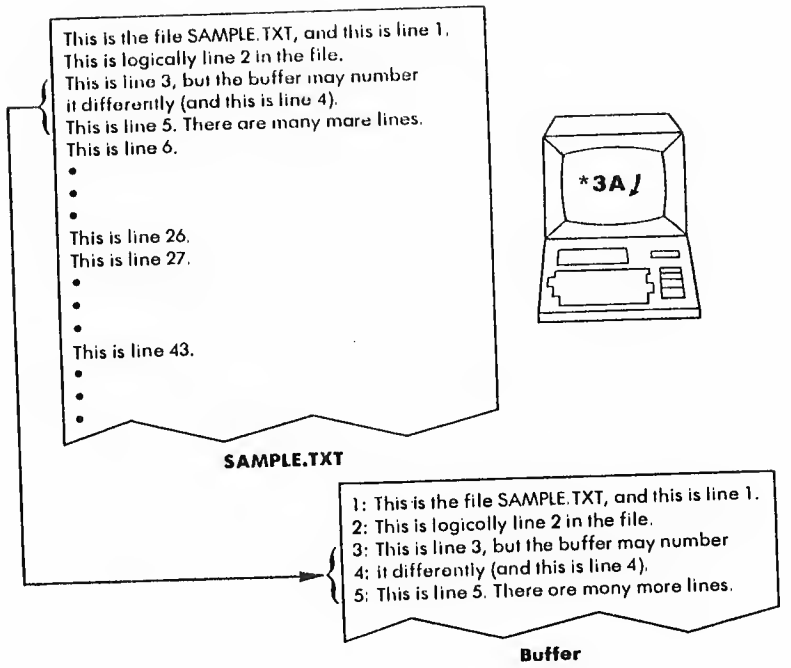


Figure 4.5: Appending Three More Lines

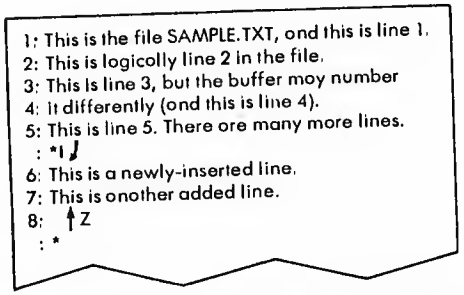


Figure 4.6: Adding Two New Lines From the Keyboard

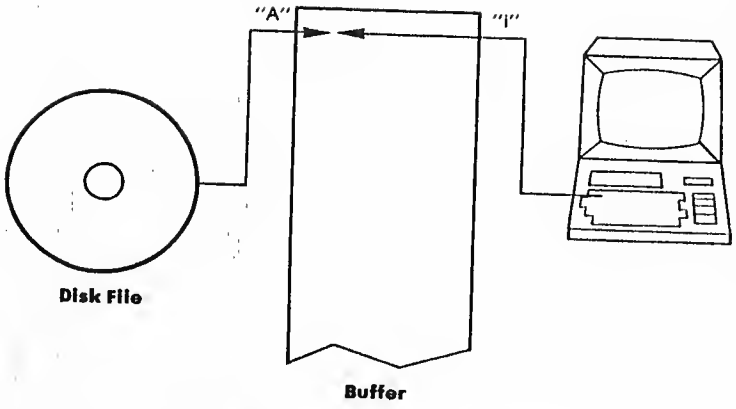


Figure 4.7: Putting Lines into the Buffer

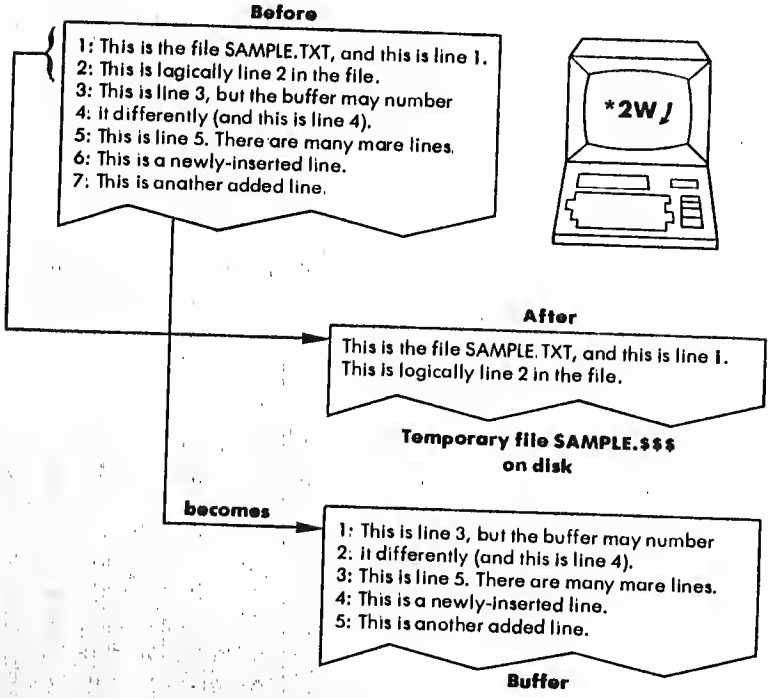


Figure 4.8: Saving the Buffer on the Disk

The example in Figure 4.8 shows a write operation of two lines. The lines in the final edit buffer move up to the beginning, releasing more space to append more lines. The next example shows an append operation of the next twenty lines.

After modifying the text in the buffer, you could end the ED session (and write the rest of the buffer to the output file) by using the E command, as shown in the example in Figure 4.9. The E command also copies the *rest* of the source file — the lines not appended to the buffer — to the temporary output file. Note that the temporary output file contains the lines of text in their proper order.

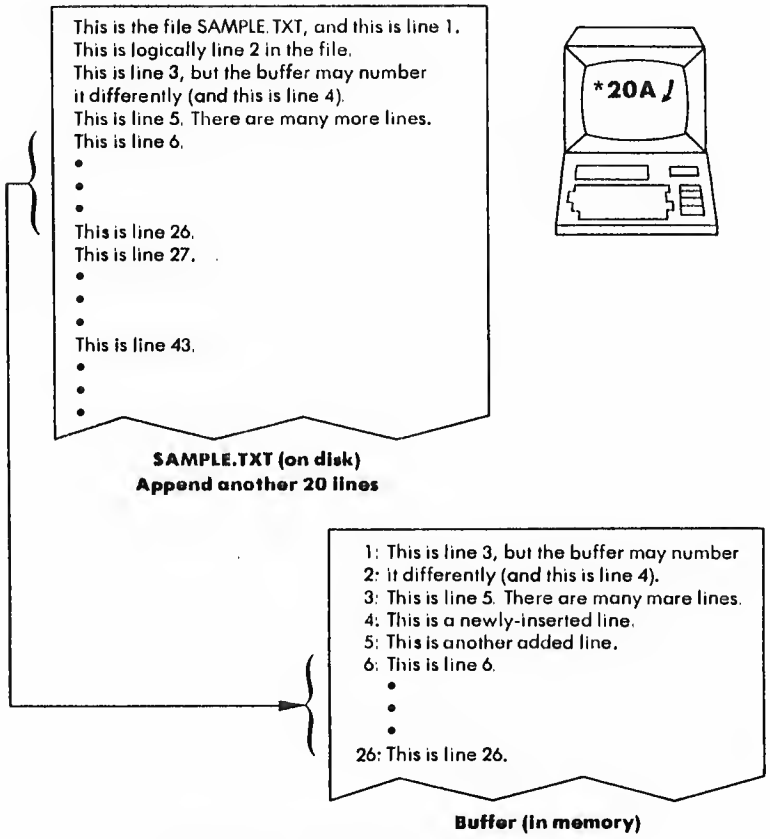


Figure 4.9: Appending 20 Lines to the Buffer

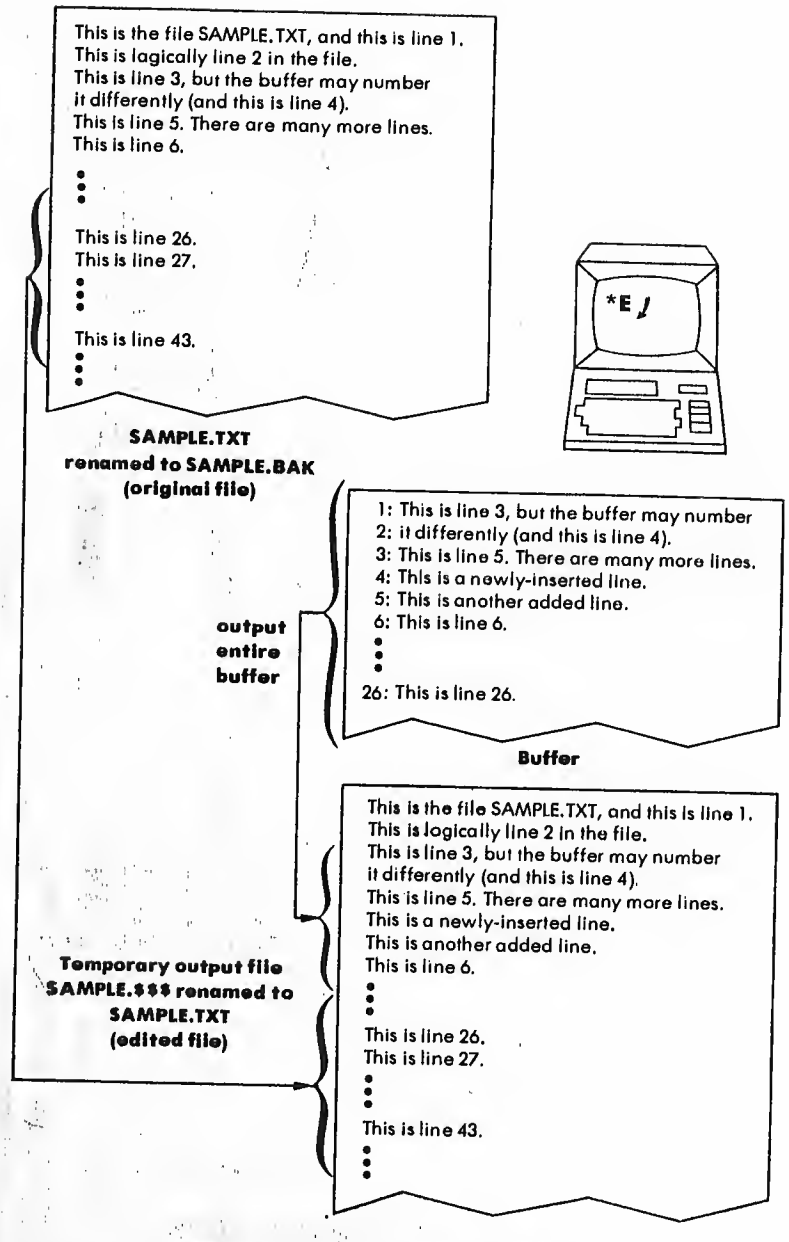


Figure 4.10: Finishing an Edit Session

As soon as the edit buffer is properly copied into the temporary output file, which is called SAMPLE.$$$, ED does two things: it renames the original SAMPLE.TXT to SAMPLE.BAK, and it renames SAMPLE.$$$ to SAMPLE.TXT. By doing this, ED creates a backup copy of the original SAMPLE.TXT (called SAMPLE.BAK), and renames the newly modified SAMPLE.$$$ to SAMPLE.TXT. (See Figure 4.10.) That is why it appears that ED modifies the file SAMPLE.TXT—actually, ED modifies the text in the edit buffer and uses SAMPLE.TXT as a backup and SAMPLE.$$$ as the future version of SAMPLE.TXT. NOTE: when you execute ED on a file, ED automatically deletes any 'BAK' file associated with the text file (in preparation for the new 'BAK' file that ED creates). The 'O' and 'Q' commands do not prevent this action, so be careful.

FILE MANAGEMENT

When you give ED a filename, it looks for the file; if ED does not find it, ED creates it. This file is called the "source" file, in CP/M's documentation. When you append text to the edit buffer from the source file (using the A command), ED copies the text lines from the top of the file into the buffer, counting the number of lines you specified in the A command. When you write lines to the temporary output file created by ED (filename.$$$), you free up space in the buffer in order to append more lines of text from the source file. As you write to the output file, lines are appended so that they stay in the same order (you can write specific lines to the output file in order to change that order). When you finish editing, or terminate ED using the E (or H) command, ED automatically renames the original file (the source file) to a file with a BAK extension (e.g., SAMPLE.BAK) to denote it as a backup file, and D renames the temporary output file (e.g., SAMPLE.$$$) to the name of your original (source) file (e.g., SAMPLE.TXT), so that your text file contains the newly-modified text.

ACCIDENTAL TERMINATION

If you terminate ED accidentally, without using the E (or H) command (i.e., there is a system error, or you inadvertently hit ↑C to reboot the system, or the power is cut off), the temporary output file (e.g., SAMPLE.$$$) will still remain with that filename, and your

original source file (e.g., SAMPLE.TXT) would still be the old copy. The $$$ file would only have text you had already written to it (so it might not be useful), the edit buffer would be lost, and the original file would be the unedited version (i.e., before you invoked ED).
NOTE: use the Q (quit) command to do this on purpose.

You can merge text lines from another text file with the text already in the edit buffer by using the R command, discussed later in this chapter (the other file is not modified in any way).

A SESSION WITH THE EDITOR

To create a new text file, first think of a name (a name that is not already used with another extension). We will use QUOTE.TXT as an example. Execute the ED command and create QUOTE.TXT by typing:

In this example, we are In drive A; therefore, QUOTE.TXT will be in drive A. We are in drive A because the ED program (ED.COM) is in drive A. The example in Chapter 1 showed how you can execute ED from another drive by prefixing the drive letter and colon before the filename ED. You can also execute ED from drive A and put QUOTE.TXT on drive B by prefixing 'B:' to 'QUOTE.TXT'.

ED will display the message 'NEWFILE' if it is creating a new file. When ED is ready for an ED command, it will display the ED '*' prompt. You can now type any ED command. To insert new text from the keyboard, you should use the I command. The I command starts inserting text immediately after the CP (character pointer). Since you have not explicitly moved the CP (with an ED command), the CP is at the beginning of the buffer. To start inserting new text, type:

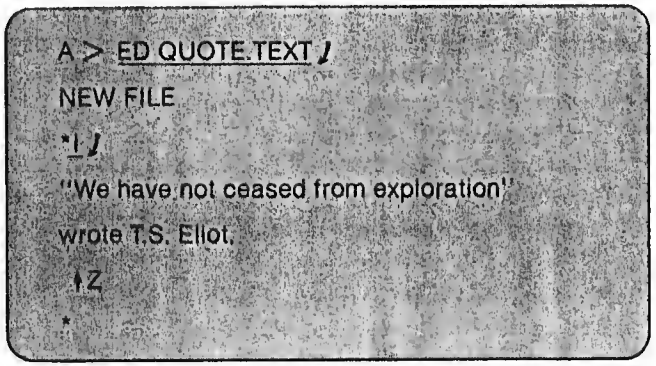
To actually insert text, you type the text as you would on a typewriter. To stop inserting text, you hold down CTRL while pressing the Z key (↑Z).

When you type 'I ↵', ED moves the cursor (the blinking pointer on

the screen, or whatever symbol your terminal display uses) to the next blank line. You can now type text, which will automatically be inserted into the edit buffer as you transmit each line. *Each line must end with a* RETURN (as with a typewriter) — RETURN (represented with a ↵ in this book) transmits the line to the edit buffer. The line then becomes a *line in the buffer* (identified by a line number). If you want to type "a line in the buffer" that is actually longer than a line you can type at your terminal, use the combination of CTRL and E (↑E) to move the cursor to the next line on your screen without transmitting the line to the edit buffer. When you finally do hit RETURN, the entire line is transmitted to the buffer. Long lines cannot exceed 128 characters.

We will start the example from the beginning again. Note that you can use the keyboard editing keys RUBOUT (DELETE), ↑U (delete the line), ↑E (return the "carriage" without transmitting the line), and ↑R (retype the last line) in both the old and new versions of ED. You can use ↑H (backspace and delete a character) and ↑X (backspace to beginning of line) in only the new version of ED.

Here is our example again, from the start:



We use the combination CTRL and Z to stop inserting text (↑Z). We now have two lines of text — the first is the quotation, and the second is 'wrote T.S. Eliot'.

If you wish, you can use the U command before the I command, and anything you type is automatically converted to UPPER case (although as you type the text, it appears to be in upper and lower case). To turn off this automatic UPPER case translating, use the -U command (negative U).

Now that we have text in the buffer, we can display it. First, we have

to move the CP (character pointer) back to the beginning of the buffer, since the I command inserted text and moved the CP. Here is an illustration showing the text and the position of CP in the buffer:

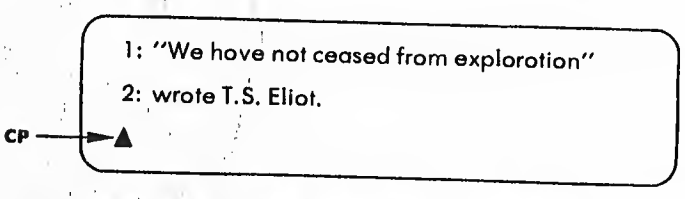


Figure 4.11: The CP is at the End of the Buffer

The manual for ED explains that the CP is *between* two characters. This is difficult to visualize and to use, so we changed our description to make it easier. The CP is an "imaginary" object, a reference pointer to use with ED commands. When you refer to Digital Research's documentation, bear in mind that their descriptions are based on CP being between two characters, and our descriptions are based on CP pointing to the rightmost of those two characters. The illustration in Figure 4.12 should clarify this:

Figure 4.12: Showing the CP's Position

Digital Research says that CP is *before* the first character of a buffer when moved to the beginning, and we say that CP is *at* the first character of a buffer when moved to the beginning.

DISPLAYING TEXT IN THE BUFFER

We will use the T command to display the text in the buffer. The format of the T command is:

±nT

where n can be zero, or any number, or a number (#) sign. This is illustrated in Figure 4.13. If n is zero, T will display the current line up to

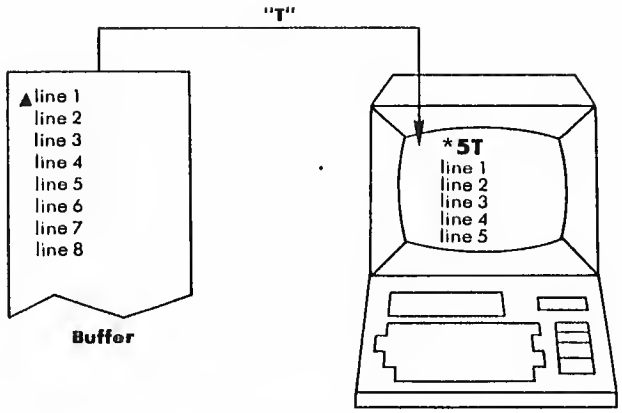


Figure 4.13: Displaying Text

(but not including) the CP (the current line is the line with CP in it). If you do not specify n, 1 is assumed. If n is 1 (or if it is assumed to be 1), the current line is displayed from the CP to the end of the line. If n is a positive number, T will display n number of lines from the CP (current line) on; if n is a negative number, T will display lines before the CP (and not including the CP). If you use the number (#) sign, T will substitute '65535' (maximum number of lines allowed in the buffer), i.e., the entire buffer will be displayed. You can type the entire buffer by moving CP to the beginning of the buffer (by using the B command, shown in Figure 4.14) and using a number (#) sign with T; other-

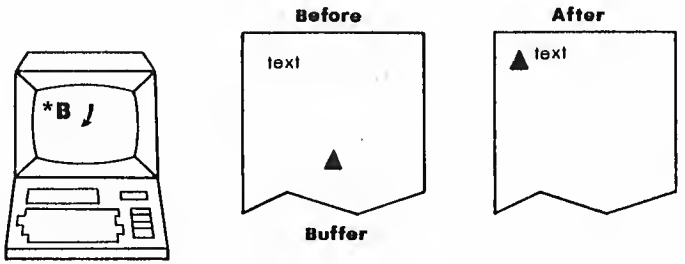


Figure 4.14: Moving the Cursor

wise, a number (#) sign will display all of the lines *following* (and including) the CP, and a negative number (−#) sign will display all of the lines previous to (but *not* including) the CP.

Here is an example:

```
*-2T↵
1: "We have not ceased from exploration"
2: wrote T.S. Eliot.
2: *
```

If your display does not include the line numbers (numbers followed by the colon), try executing the V command (version 1.4 of CP/M or later):

```
*V↵
:*
```

The ':' tells you that the CP is at the beginning of line 3; however, line 3 has no text yet, so the CP is actually pointing to the end of line 2 (after the 'Carriage Return' sequence).

The command '-2T' above tells ED to display only the two lines before the one containing the CP (i.e., the current line, which is line 3).

In CP/M versions 1.4 or later, you can specify the actual line number in a T command to display that line. For example, if you just wanted to display line 1, you would type '1:T' as a command:

```
2: *1:T↵
1: "We have not ceased from exploration"
1:
```

Note that the command '1:' with 'T' moved the CP to line 1 (current line is now line 1). You can specify a line number as a command to move

the CP to that line. For example:

```
1: *2:↵
2: *
```

A simple T command will display the current line:

```
2: *T↵
2: wrote T.S. Eliot.
2: *
```

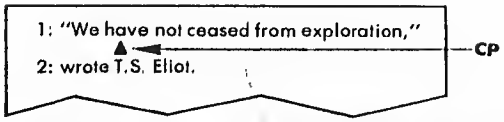
An easy way to display the entire buffer is to execute two commands: the B command to move the CP to the beginning of the buffer, and the T command with a number (#) sign to display the entire buffer. Note that you can put both commands on the same command line and execute them:

```
1: *B#T↵
1: "We have not ceased from exploration"
2: wrote T.S. Eliot.
1: *
```

In the above examples, the CP pointed to the first character in the line. You can move the CP in the current line by using the C command:

±nC

C will move the CP +n characters forward, or −n characters backward. For example, if the CP is at the beginning of line 1, and you type this command:




```
1: *5C
1: *
```

The CP would be pointing to the sixth character ('a', since a quote counts as one character) in line 1. The command 'T' would display the line from the CP to the end (including the CP):

```
1: *T
ave not ceased from exploration"
1: *
```

The command '0T' would display line 1 from the beginning up to but *not* including the CP:

```
1: *0T
"We h
1: *
```

SAVING THE FILE AND ENDING THE ED SESSION

At this point, you should learn how to save the contents of the buffer and leave ED. The easiest way to save and exit ED at the same time is to use the E command:

```
1: *E
```


Wherever you are in the buffer, this command will empty the buffer (and the rest of the source file, if you had not appended the entire source file) into the temporary output file and rename the output file to the name of your source (original) file. If you started with an empty QUOTE.TXT and added the text as shown in the previous examples, you should end up with a QUOTE.TXT containing a line by T.S. Eliot, and a QUOTE.BAK that is empty (a backup of the file before you edited it).

APPENDING LINES TO THE BUFFER (EDITING AN EXISTING FILE)

When you have an existing text file, and you execute ED to edit it, the A command then has to be executed to bring text lines into the edit buffer. Otherwise, there will be no text in the buffer except the text that you insert by using ED's I command. You might want to insert new text *before* the first line of your existing text file—by using the I command to insert new text before appending the existing file's text to the buffer with the A command; however, you can easily do this *after* the old text is in the buffer. So, you will want to use the A command first to see the old text. The A command adds ("appends") lines to the end of the buffer.

The A command's format is:

nA

The command A will append n lines of text from the original (source) text file. If you do not specify n, A appends only one line. If you use a number (#) sign instead of n, A will bring the entire source file (up to 65535 lines) into the edit buffer. Since most text files are not that large, you can use the form '#A' to bring in any file that would probably fit in the edit buffer. If you specify a zero for n, the A command will append to fill *half* of the buffer (useful for large files). Use the form '0A' in conjunction with '0W' to append and write-out half of the buffer (discussed in this chapter in "Writing Lines Out to the File").

If you append only a portion of your source (original) file to the buffer, your *next* A command would start appending from the source file where the last A left off. You would easily append ten lines, change some text, write out the ten lines to the temporary output file (with the W command discussed later), and append more lines from the source file. Or, if you have enough room in the buffer, you could append more

lines from the source file to the end of the existing lines in the buffer.
Here is a simple example, using QUOTE.TXT as a source file:

```
A>ED QUOTE.TXT ↵
*V ↵  (if using version 1.4, not needed in version 2.2)
:*A ↵  (appends only one line)
1:*2A ↵  (appends next two lines, but there is only one
          more line in the file anyway)
2:*B#T ↵  (go to beginning and display all lines)
1:"We have not ceased from exploration"
2:wrote T.S. Eliot
1:*
```

Since there are only two lines in QUOTE.TXT, a simple '#A' command would suffice to bring in all of the lines.

MOVING AROUND IN THE BUFFER

With text in the buffer, you can move around at will and change any text, as well as insert new text anywhere. You can also locate and substitute pieces of text, and append other lines of text from a special *library source* file (discussed later).

If you are following the examples and editing QUOTE.TXT, you should now go the bottom of the buffer and insert more text. The -B (negative B) command is used to move to the end of the last line, as in this example:

```
1: "We have not ceased from exploration,"
2: wrote T.S. Eliot,
:
▲
```

Buffer

```
:*-B ↵
:*
```


The "end of the last line is actually a 'Carriage Return', which in ASCII code is a combination of RETURN and LINE FEED: two characters that perform the operation of returning the carriage and generating a new line. Therefore, the "end" of the last line is actually the beginning of the next new line; however, the next new line's number is not displayed until you insert characters using the I command. You can insert characters that will form the new line. For example:

```
1: "We have not ceased from exploration"
2: wrote T.S. Eliot,
3: "And the end of all our exploring,
4: will be to arrive where we started,
5: and know the place for the first time."
:
▲
```

Buffer

```
:*I ↵
3: "And the end of all our exploring, ↵
4: will be to arrive where we started, ↵
5: and know the place for the first time." ↵
6: ↑Z ↵
:*
```

Now you can display the entire buffer by using the B and T commands:

```
:*B#T ↵
1: "We have not ceased from exploration"
2: wrote T.S. Eliot,
3: "And the end of all our exploring,
4: will be to arrive where we started,
5: and know the place for the first time."
1:*
```

The easiest way to move to another line is to use line numbers. If you have version 2.2 of CP/M, ED will automatically display line numbers. If you have version 1.4, this feature is turned off; turn it on by executing the ED command V (-V turns it off again). If you have a version earlier than 1.4, you are out of luck—you cannot display line numbers and must use the L command to move to another line.

To select a line, type the line number followed by a colon as an ED command:

```
1: *2:↵
2: *
```

To select a *range* of lines, type the first line number, followed by *two* colons and the second line number, as in this example (the T command can also be used to display the range of lines):

```
1: "We have not ceased from exploration"
2: wrote T.S. Eliot,
3: "And the end of all our exploring,
4: will be to arrive where we started,
5: and know the place for the first time.
```

Buffer

```
2: *3::5T↵
3: "And the end of all our exploring,
4: will be to arrive where we started,
5: and know the place for the first time."
3: *
```

When you specify a single line, the CP is moved to the beginning of that line. When you specify a range of lines, the CP is moved to the beginning of the first line in the range, and ED counts the number of lines

in the range (in this case, three), and applies that number to the T command (i.e., '3T') to display the range. The CP remains on line 3.

You can also move up and down lines by using the L command. The format for the L command is:

±nL

The L command will move the CP +n lines forward or -n lines backward in the buffer, and put the CP at the beginning of the line selected. Here is an example:

```
3: *1L↵
4: *-2L↵
2: *
```

The form '0L' (where n is zero) moves the CP to the beginning of the current line.

You can use line numbers to select a range of lines that begin at the current line by preceding the ending line number with a colon:

```
2: *:5T↵
2: wrote T.S. Eliot,
3: "And the end of all our exploring,
4: will be to arrive where we started,
5: and know the place for the first time."
```

CHANGING, INSERTING AND DELETING TEXT

Text in a line is changed by moving the CP to the text and deleting existing text and inserting new text. You can also find a group of characters and substitute another group, as discussed in the next section.

When you delete a line of text, the line numbers reflect the deletion

and change accordingly, as in this example:

```
2:*1::5T↵
1:"We have not ceased from exploration"
2:wrote T. S. Eliot
3:"And the end of all our exploring,
4:will be to arrive where we started,
5:and know the place for the first time."
1:2↵
2:K↵
2:*1::4T↵
1:"We have not ceased from exploration"
2:"And the end of all our exploring,
3:will be to arrive where we started,
4:and know the place for the first time."
1:*
```

The K command above deletes line 2, and the other lines "move up" to use the space. The opposite occurs when you insert a line — the other lines "move down" to accommodate the newly-inserted line.

The format of the K command is:

+nK

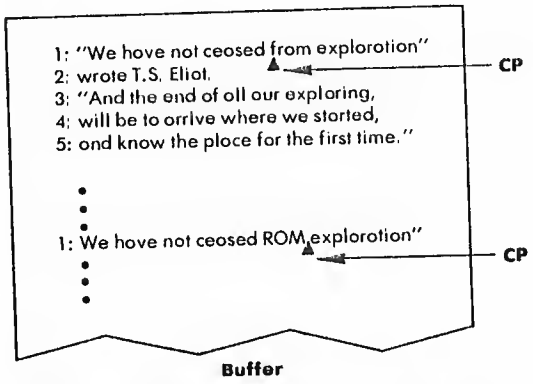
The K command will delete ("kill") the current line if no n is specified. Otherwise, the K command will delete either +n or −n lines from the current line. If you supply a number (#) sign for n, K deletes 65535 lines following (and including) the current line (+#K), or 65535 lines behind (and not including) the current line (-#K). Use these forms to clear out unwanted lines from your buffer, but remember — you cannot bring back those lines (unless they already exist in the source or backup file).

To delete *characters* (not entire lines), you use the D command:

+*n*D

The D command deletes (erases) the character pointed to by the CP (character pointer) if no n is specified. Otherwise, the D command deletes +n characters following (and including) the CP, or −n characters behind (and *not* including) the CP.

Here is an example: we will change the word 'from' to 'ROM' (and then back to 'from'). We will do it the hard way — moving the CP by using the C command, deleting 'from' by using the D command, and inserting 'ROM' by using the I command:




```
1: *1::4T↵
1: "We have not ceased from exploration"
2: "And the end of all our exploring,
3: will be to arrive where we started,
4: and know the place for the first time."
1: *T↵
1: "We have not ceased from exploration"
1: *2OC↵
1: *4DIROM↑Z0LT↵
1: "We have not ceased ROM exploration"
1: *
```

In this example, we move the CP to the 21st character by using the command '20C'. Counting the character pointed to by CP, we delete 4 characters using the D command. Then we use the I command to insert 'ROM' (terminated by ↑Z)—note that the I command inserts before the CP and moves the CP. The D command deletes the CP character and also moves the CP. We use the L command to move the CP to the beginning of the line, and we use the T command to display the line.

This is a new form of the I command:

*I*insertion↑Z

It performs the same function as I ↵ except that it does not automatically insert carriage returns to generate new lines. If you are going to change a group of characters including a 'Carriage Return' sequence, use the S command (substitute) described in the next section, because it is much easier than counting characters in order to move the CP properly.

FINDING AND SUBSTITUTING TEXT

An easier way to move the CP to a group of characters in the text is to "find" that group of characters by using the F command. The S command, discussed next, may be used to "find and substitute." The F command is a primitive version of the S command, and will be described first.

An example will show the format for the F command. We will execute F to find 'ROM' in line 1:

```
1:*FROM↵
1:*
```

The illustration shows where the CP is at this moment:

```
"We have not ceased ROM ▲ exploration"
                        CP
```

The F command moves the CP to the character immediately following the last character found, in order to make it easier to use the D command (to delete the characters found). We will move CP back to the beginning of the line, and execute the F command along with the D and

I commands in the next example:

```
1:0L↵
1:*FROM↑Z-3DIfrom↑Z0LT↵
1:"We have not ceased from exploration"
1:*
```

-3 characters from the CP (i.e., 'MOR'). The CP is now in position for us to use the I command to insert "from." The '0LT' command combination moves the CP to the beginning of the line and displays the line.

Note that you can end an F command with a RETURN if you are not going to include another command on the same line.

The F command can also begin with a number (i.e., nF where n is a positive number) that will tell it to find n occurrences of the group of characters. For example, if you type the command '5Fthis ↵', the command will not stop or move the CP until it finds the fifth occurrence of 'this'. The F command can search the text that is in the buffer; in order to search the *entire source file*, you have to use the N command (discussed in "Advanced ED Operations").

If you are trying to find a group of characters including the Carriage Return sequence, you can substitute a special key combination, ↑L, to represent the sequence of RETURN and LINE FEED. In our example with the S command, we will demonstrate the use of ↑L.

The S (substitute) command is the one most often used to find and substitute groups of characters. The S command combines the actions of the F, D, and I commands shown earlier. The format for the S command is:

$$n\text{Soldtext}\uparrow\text{Znewtext}\left\{\begin{matrix}\uparrow Z\\ \leftarrow\end{matrix}\right\}$$

The S command searches the buffer for the group of characters *oldtext* and substitutes the group of characters *newtext*. You can use ↑Z or RETURN to end the *newtext* string (use ↑Z if you want to add another command to the S command). You can execute this substitution *n* number of times, and it will execute until it reaches the *n*th time, or until it reaches the end of the buffer.

For example, if you typed the command "#SPeking ZBeijing ↵", it would substitute the word "Beijing" for "Peking" throughout the text in the buffer (the number sign (#) represents 65535 times).

We will use the S command to alter our example and change the text from prose to poetry. First, we will correct the end of the first line and the beginning of the second line in one S substitution:

```
1:*2T↵
1:"We have not ceased from exploration"
2:"And the end of all our exploring,
1:*S"↵L"↵Z/↵L↵ZB2T↵
1:"We have not ceased from exploration/
2:And the end of all our exploring,
1:*
```

In the above example, we use the S command to find the group of characters starting with unquote, and ending with a quote on the next line (↵L stands for a Carriage Return), and substitute a slash followed by a Carriage Return. We then execute the combination 'B2T' to move the CP to the beginning of the buffer and display the next two lines.

Our next example will substitute a slash (/) for a comma (,) in two lines:

```
1:*2,4T↵
2:And the end of all our exploring,
3:will be to arrive where we started,
4:and know the place for the first time."
4:*2↵
2:*2S,↵Z/↵ZB4T↵
1:"We have not ceased from exploration/
2:And the end of all our exploring/
3:will be to arrive where we started/
4:and know the place for the first time."
1:*
```


In this example, we first move the CP to the beginning of line 2 ('2:'), and then perform the substitution ('/' for ','). Then we move the CP to the beginning of the buffer and display four lines (B4T).

We need to put only the first characters of lines 3 and 4 into upper case:

```
1:*3:↵
3:*DIW↵Z1LDIA↵ZB4T↵
1:"We have not ceased from exploration/
2:And the end of all our exploring/
3:Will be to arrive where we started/
4:And know the place for the first time."
1:*
```

In this example, we move the CP to the beginning of line 3 and delete the first character (D command deletes 'w'). Then we insert (I command) a capital W. We move the CP to the next line (1L command), delete the first character (D command deletes 'a'), and insert a capital A. Finally, we move the CP to the beginning and display four lines.

WRITING LINES OUT TO THE FILE

Normally, you would end your ED session by using the E or H commands. Each performs a write operation on the entire buffer; i.e., it *writes* the lines of text to the temporary output file. Both commands also copy the rest of the source file (i.e., the lines that were not appended to the buffer) to the output file, and rename the files so that you end up with the newly-modified text in the source file. The E command also terminates ED; whereas the H command keeps you in ED and prepares the new source file for further editing.

If you just want to write lines to the output file without copying the entire buffer or the rest of the source file, use the W (write) command. The W command takes the following form:

```
nW
```

The W command writes the next n lines from the current line (including the current line) to the output file. The output file is the temporary file with '$$$' as an extension. If you do not specify n, the current line is written out to the file.

To illustrate the W command, we will add more lines to our example and write several lines out to the temporary output file:

```
1:*-BI↓
5:↓
6:                              — wrote T.S. Eliot ↓
7:↑Z
6:B#T↓
1:"We have not ceased from exploration/
2:And the end of all our exploring/
3:Will be to arrive where we started/
4:And know the place for the first time."
5:
6:                              — wrote T.S. Eliot
1:*3W↓
1:#T↓
1:And know the place for the first time."
2:
3:                              — wrote T.S. Eliot
1:*#W↓
*
```

In this example, we first add new text to the bottom ('-BI' commands) of the buffer (use a RETURN to make a blank line, and an I to insert a tab space). Then we display the entire buffer with the CP at the beginning of the buffer ('B#T' commands). We then output the next three

lines, including the current line (lines 1, 2, and 3). These lines are now in QUOTE.$$$, the temporary output file. The remaining lines of the buffer are now renumbered. The last command ('#W') outputs the next 65535 lines including the current line, which outputs the remainder of the buffer. The buffer is empty, and ED displays no line number (':*').

We still have to save the *rest* of the source file (if there were any unappended lines) and *rename* the temporary output file QUOTE.$$$ to QUOTE.TXT (and rename the old QUOTE.TXT to QUOTE.BAK). The E and H commands will do these operations automatically.

A special form of the W command goes with the '0A' form of the A command — '0W' (where *n* is zero) will write-out half of the buffer, and '0A' will append to fill half of the buffer. The number of lines that equal half of the buffer depends upon the length of your text lines and the size of your system. These forms are mostly used to "pull in half a buffer, push out half, pull in another half, etc."

If you want to rearrange lines in the buffer, or interweave lines from another file, then you must use special commands (described in "Advanced Operations") to bring lines from files to the buffer, and *then* write the finished version out to a file by using the W command (or, save the entire buffer and the remainder of the source file with the E or H commands).

ADVANCED ED OPERATIONS

Searching Through the Source File

You can use the N command to do a "find" (F command) on the buffer *and* the rest of the source file. The N command operates in the same manner as the F command but does not stop at the end of the buffer — it performs an automatic append and continues to append lines from the source file until it finds the group of characters that you specified in the command.

The format for the N command is:

nNtext { ↑Z / ↓ }

NOTE: braces { } denote an option between two commands. The N command searches for the *n*th occurrence of *text* in the lines following the CP in the buffer; if it does not find the *n*th occurrence, it will append

lines from the source file into the buffer until it does. You can end your *text* string with ↑Z if you are going to add another command; otherwise, you can use RETURN.

The N command will put the CP after the last character of the *n*th occurrence of *text*, just like in F commands.

Inserting Text From a Library Source File

A "Library Source File" is a convenient term in Digital Researeh's documentation for any file with a 'LIB' extension (e.g., SAMPLE.LIB). You can use a "library source file" as an alternate source file—a file with text that you want to insert into the buffer to merge with lines from the original source file. To do this, you must first have a file with a 'LIB' extension that already has the text in it.

The format for the R command is:

Rfilename

The R command inserts the lines from the LIB file that you specify in *filename*. It will insert the lines at the current position of the CP in a way similar to the I command. R will insert the entire 'LIB' file until it finds the end-of-file (↑Z) marker.

Transferring Lines To and Inserting Lines from a Temporary File

If you have version 1.4 or a successive version of CP/M, you can use the new X command to transfer lines from the buffer to a temporary "holding" file, and you can use the R command to insert the lines into the buffer from this "holding" file. The "holding" file is named X$$$$$$$.LIB and only exists while the ED program is running. Any normal termination of ED will delete the file, but if you terminate ED with a ↑C (warm boot), the file will still be there (however, once you execute ED again, the file is deleted).

The format of the X command is:

nX

The X command transfers (copies) the next n lines from the current line to the temporary file X$$$$$$$.LIB. The lines in the buffer will remain there: they are only copied to the temporary file. If you wish, you can use the K command to delete the lines after copying them. The transferred lines accumulate in the temporary file in the order in which they

are copied by successive X commands. Using this command, new text may be built in successive blocks.

The lines can be retrieved by using the R command in the form 'R' without the *filename*. All of the transferred lines are then inserted following the CP (similar to the I command). However, the R command does not empty the temporary file, it simply copies the lines. You can retrieve them again and again (useful for repetitious lines). You can empty the temporary file (i.e., delete it) by executing the form '0X' (where n is zero) of the X command.

If you want to preserve the temporary file X$$$$$$$.LIB, use ↑C to abort ED, and immediately rename the file (to get rid of the LIB extension) so that it won't be destroyed when you execute ED again.

Juxtaposition

The J command is used to juxtapose three groups of characters in a text. It finds a group of text, juxtaposes a second group, and deletes the characters following this pair until it finds yet a third group of text, effectively juxtaposing all three groups of text.

The format for the J command is:

nJstring1↑Zstring2↑Zstring3 {↑Z / J}

The J command starts searching from the CP in the buffer for the first occurrence of 'string1.' If it finds 'string1,' the J command inserts string 2 immediately following 'string1', and moves the CP to the end of 'string2'. The J command then looks for 'string3'; if it finds 'string3', the J command deletes all characters between 'string2' and 'string3', and leaves the CP pointing to the first character of string3. If the J command does not find 'string3', it doesn't delete anything. The J command does this n number of times, or until it runs out of lines in the buffer.

One use of the J command is to shorten lines of text. Pick a group of characters that will be the end of the line to be shortened. They will be the 'string1'. The concept is to juxtapose them to the Carriage Return sequence, represented by ↑L ('string3'). This result is achieved by inserting a blank or null 'string2'.

Repeating a Set of Commands

You can string together several ED commands into one "command" that can then be executed repeatedly. The M command (for "macro") takes the following form:

nMstringofcommands $\left\{\begin{matrix}\uparrow Z \\ / \end{matrix}\right\}$

Group your commands into 'stringofcommands', preceeded by 'M', and M will execute the commands n times, if n is greater than one. If n is zero or one, the 'stringofcommands' are executed repeatedly until an error occurs (like reaching the end of the buffer).

Here is an example that changes all occurrences of 'Peking' to 'Beijing' within the current buffer, and dispatches each line that is changed:

MSPeking↑ZBeijing↑ZOTT

ED'S ERROR CONDITIONS

When an error occurs within the ED program, ED displays the last character it saw before the error, and one of the following error indicators:

?	Dan't recognize the cammand, what is it?
>	The buffer is full. Use one of the commands D, K, S, W, E, ar H to remove characters. Or, your string with F, N, ar S is toa long.
#	Cannat execute the cammand that many times (os in on F cammond reoching the end of o buffer).
0	Cannat apen the LIB file in on R command (check to see if the LIB file exists, or if yau used the right filenome).

Figure 4.15: ED's Error Messages

Newer versions of CP/M display 'BREAK x AT c' where x is one of the error symbols, and c is the ED command that was executing.

Occasionally, the system behind ED (i.e., CP/M or MP/M) detects a system error condition (such as a disk error). Depending upon the condition, you can usually terminate ED by doing a ↑C (warm start), but you must first retrieve the original copy of the source file. For example, if CP/M detects a CRC (cyclic redundancy check) error, it will display:

PERM ERR DISK d

where d is your current disk drive. You can choose to ignore the error by typing any character at your terminal (however, you should check your buffer for mistakes), or you can perform a warm boot (↑C) or a cold boot (reboot the system) and retrieve the BAK file (file with the BAK extension, e.g., QUOTE.BAK).

CP/M Version 2.2 Enhancements Summary

In CP/M version 2.2, ED assumes that the line numbering option is always on. To eliminate the line numbers, type: −V. (Of course, this mode can be reinstated with : V.)

When the insert mode is used (I), the usual CP/M control characters may all be used: DEL, ↑C, ↑E, ↑H, ↑J, ↑M, ↑R, ↑U, ↑X. They are described in Appendix D.

Finally, ED respects the file attributes of version 2.2. For example, a read/only file may be examined, but not changed. If this file must be modified, then its read/only status must first be changed to R/W, by using a STAT command.

SUMMARY

In this chapter, you have learned how to use an editor, ED. ED is a general text editor that allows you to modify text with just a few key strokes. ED is used most often to create a simple file, such as a program (if no other specialized editor is available). Although ED's commands are less convenient than a specialized word processor, it can be used to type letters or documents.

ED is a convenient tool for correcting and modifying existing files. (In particular, ED can be used to "clean up" a System Diskette that has erroneous or spurious characters due to mishandling.) ED can also be

used to substitute new words or lines into a file. Summaries of ED's control characters and commands are presented in Appendices D and E.

In this chapter, you have not only learned how to use ED, the editor, but you have learned to use many commands, control characters and other conventions. You have also learned how to operate on files, and have followed transfers of text between disk, memory and CRT. This knowledge will help you understand the operation of most other programs on the computer.

5

INSIDE CP/M (AND MP/M)

INTRODUCTION

This chapter will describe the internal operation of the CP/M operating system. You will want to read this chapter if you are interested in learning about the ways in which an operating system works, or plan to modify or use some of CP/M's routines. However, if you only wish to use CP/M to accomplish a specific task, the information presented in this chapter is not necessary.

This chapter will be of greatest value to a systems programmer, or any person already familiar with programming who wants to understand how CP/M operates. Since a book on CP/M would not be complete without this information, this chapter is presented for these specific readers. If you want to know "what is happening under the hood," read on.

We will first present a simplified overview of CP/M's operation, by introducing its logical components, their roles, and the way they interact with each other. After that the *memory allocation,* i.e., the way in which these software modules are spread over the memory will be described, as well as the organization of the file system. Then, each of CP/M's three modules will be presented in detail along with the commands that are provided to use their capabilities. Another section will discuss the problems encountered when adapting CP/M to new hardware configurations and an example of CP/M alterations to make the system behave as a menu-driven system, will be shown. Finally, a special section will be devoted to MP/M.

APPENDIX D

ED CONTROL CHARACTERS

Keys	Meaning
CTRL-C (↑C)	System restart (warm boot), restores system prompt.
CTRL-E (↑E)	Moves cursor to next line to continue command line (without executing or transmitting line).
*CTRL-H (↑H)	*Backspaces cursor to erase last character typed.
CTRL-I (↑I)	Moves cursor a "tab" space (7 columns long).
*CTRL-J (↑J)	*Performs a RETURN.
*CTRL-M (↑M)	*Performs a RETURN.
CTRL-L (↑L)	Replacement for Carriage Return sequence generated by RETURN in strings used with search and substitute commands.
*CTRL-R (↑R)	Retype current line (types a clean line).
CTRL-U (↑U)	Delete current line.
*CTRL-X (↑X)	*Backspace to beginning of current line and erase line.
**CTRL-D (↑D)	**Detach the current program from the terminal.
RETURN (↵)	Transmit (execute) the current line, or generate a Carriage Return to separate lines of text file.
RUBOUT or DELETE	Delete the last character typed (echoes the character).
CTRL-Z (↑Z)	Terminate the I command's inserting, or separate strings of text in search and substitutions, or place as marker at end of text file.
CTRL-P (↑P)	Echo everything typed or displayed at the lineprinter.
CTRL-S (↑S)	Temporarily halt a long display (strike any key to continue display).
BREAK	Discontinue execution of currently-executing ED command.

*CP/M 2.2
**MP/M

APPENDIX E

ED COMMANDS

Commands	Meaning
nA	Append n lines (or 1 line if no n) from the source file to the edit buffer. A "#" for n will append 65535 lines (fill the buffer), and o zero for n will append to fill half of the buffer (number of lines depends on size of your buffer and your system).
+/−nB	Move CP* to beginning if +B, or end if −B of the buffer.
+/−nC	Move CP forward (+) n characters or backword (−) n characters. The CP counts o "carriage return" sequence as two characters (RETURN and LINE FEED).
+/−nD	Delete n characters forward (+), including the CP, or delete n characters backward (−), not including the CP. If no n, delete only the character pointed to by CP.
E	End ED session normally. The E command saves the buffered text ond the rest of the source file text in a temporary output file, then renomes the output file to the name of the source file (while copying the source file into a backup ".BAK" file to preserve the original source file). ED then terminotes, bringing back the system.
nFstring {↑Z / ↵}	Find the string of characters n times (if no *n* is specified, find it once). F searches after the CP in the buffer ond moves CP to the end of the found string. Follow string with Z terminator if you will add more ED commonds; otherwise, use RETURN to terminate string.
H	End ED session, perform an E command, then execute ED again on the new source file (save your file updotes and return to edit again).

NOTE: you can substitute o "#" character for n in any of the ED commonds; it gives n the highest value it can hove −65535.

*CP is the character pointer.

I↵	Insert a new line of text after the CP, moving CP to the end of last line inserted. In CP/M 2.2: — If upper case I is used in the command, all text inserted will be in upper case only. — If lower case I is used, then the text will be inserted in upper and lower case.
Itext ↑Z	Insert characters after the CP, moving CP to the end of last character inserted.
nJstring1 ↑ Zstring2 ↑ Zstring3 {↑Z / ↵}	Juxtapose strings of text by finding string1, inserting string2 at the end of string1, and delete all characters up to but no including string3 (juxtapose all three strings of text). CP is moved to the beginning of string3.
+/−nK	"Kill" (delete) the following (+) n lines, including the CP and characters following it on the current line, or delete the previous (−) n lines including the characters behind the CP on the current line.
+/−nL	Move CP to the beginning of the current line if n is zero; otherwise, move CP to the beginning of the current line and move it forward (+) n lines or backward (−) n lines.
nMstring {↑Z / ↵}	Repeat execution of the string of ED commands n times, if n is greater than 1. If n is zero or one, M will execute the string of commands repeatedly until an error occurs.
nNtext {↑Z / ↵}	Search for the nth occurrence of text throughout the buffer and source file (terminate text with RETURN or Z to add more commands). N moves the CP to the end of the found text. N will append source lines until it finds text.
O	Omit ED session and return to original source file.

+/−nP	Move CP, display and print pages of existing text in buffer. The n is the number of pages (24 lines per page) printed, where +n prints n pages following the CP, and −n prints n pages prior to the CP. 0P (zero for n) will print the current line and page (first 23 lines following the current line). CP is moved to beginning of the printed page.
Q	Quit with no file alterations (leave the temporary file, source file and buffer file intact). Q returns you to the system. The ".BAK" backup file for the source file is not created, but if there was a previous ".BAK"file of the same name, it is deleted. (Watch out for this!)
R	Read from the file X$$$$$$$.LIB and insert the lines following the CP, moving the CP to the end of the inserted lines (does not empty the ".LIB" file).
Rfilename	Read from the filename .LIB and insert the lines following the CP, moving the CP to the end of the inserted lines (does not empty the ".LIB" file).
nSoldtext ↑ Znewtext {↑Z / ↵}	Find oldtext in the buffer following CP and substitute newtext for it; repeat the sequence n times if n is greater than one.
+/−nT	If n is not specified, or if n is 1, type (display) the characters following the CP to the end of the line. If n is zero, type (display) the characters on the current line up to but not including the CP. If n is positive (+), display the following n lines including the current line. If n is negative (−), display the previous n lines not including the current line, and display the characters on the current line up to but not including the CP. The command sequence "B#T" will display the entire buffer.
+/−U	Translate all lower case characters input (typed in or inserted) to UPPER CASE if +U, or turn off translation by executing −U.
V	Turn an line number display (line numbers are not actually in the file) for lines in the buffer.

*CP is the character pointer.

0V	Display the number of free bytes left in the buffer and the total memory size of the buffer (in decimal numbers). For example, in the display "27648/28832", "27648" are the number of bytes free, and "28832" are the number of bytes, total, in the current buffer.
nW	Write out to the temporary output file with the ".$$$" extension the following n lines from the CP (including the current line). If no n, write out only the current line.
nX	Copies the following n lines of text to the file X$$$$$$$.LIB (does not delete original lines). Retrieve lines using R command. If n is zero, this command will *delete* the file X$$$$$$$.LIB.
**nZ	Suspend the ED program for n clock ticks (approximately n seconds).
+/−n	Perform a "+/−nLT" command sequence.
n:	Move CP to beginning of line number n.
n1::n2	Specify a range of line numbers beginning with n1 and ending with n2. If either n1 or n2 is missing, substitute for it "the current line."
*CP is the character pointer.	

APPENDIX F

PIP DEVICE NAMES

LOGICAL DEVICES

CON:	for "console" or terminal, including keyboard and display (Input/Output)
RDR:	for paper tape or card reader (input only)
PUN:	for paper tape or card punch (output only)
LST:	for "listing" device like a line printer (output only).

PHYSICAL DEVICES

TTY:	for a console or terminal, a reader, a punch, or a list device. (teletype)
CRT:	for a console or terminal, or list device (Cathode Ray Tube).
PTR:	for a paper tape or card reader device.
PTP:	for a paper tape or card punch device.
LPT:	for a list device (line printer).
UC1:	for a user-defined console or terminal.
UR1:	for a user-defined reader.
UR2:	for a second user-defined reader.
UP1:	for a user-defined output (punch) device.
UP2:	for a second user-defined output (punch) device.
UL1:	for a user-defined listing device.
NOTE:	BAT: is not included, since it only re-assigns the values for RDR: and LST: (see "Assigning Devices").