Introduction to MicroEMACS

This section introduces MicroEMACS, the interactive screen editor for COHERENT.

What is MicroEMACS?

MicroEMACS is an interactive screen editor. An *editor* lets you type text into your computer, name it, store it, and recall it later for editing. *Interactive* means that MicroEMACS accepts an editing command, executes it, displays the results for you immediately, then waits for your next command. *Screen* means that you can use nearly the entire screen of your terminal as a writing surface: you can move your cursor up, down, and around your screen to create or change text, much as you move your pen up, down, and around a piece of paper.

These features, plus the others that will be described in the course of this tutorial, make MicroEMACS powerful yet easy to use. You can use MicroEMACS to create or change computer programs or any type of text file.

This version of MicroEMACS was developed by Mark Williams Company from the public-domain program written by David G. Conroy. This tutorial is based on the descriptions in his essay *MicroEMACS: Reasonable Display Editing in Little Computers*. MicroEMACS is derived from the mainframe display editor EMACS, created by Richard Stallman.

For a summary of MicroEMACS and its commands, see the entry for **me** in the Lexicon.

Keystrokes: <ctrl>, <esc>

The MicroEMACS commands use **control** characters and **meta** characters. Control characters use the *control* key, which is marked **Control** or **ctrl** on your keyboard. Meta characters use the *escape* key, which is marked **Esc**.

Control works like the *shift* key: you hold it down *while* you strike the other key. This tutorial represent it with a hyphen; for example, pressing the control key and the letter 'X' key simultaneously will be shown as follows:

```
<ctrl-X>
```

The **esc** key, on the other hand, works like an ordinary character. You strike it first, *then* strike the letter character you want. This tutorial does *not* represent the *Escape* codes with a hyphen; for example, it represents **escape X** as:

```
<esc>X
```

Becoming Acquainted with MicroEMACS

Now you are ready for a few simple exercises that will help you get a feel for how MicroEMACS works.

To begin, type the following command to COHERENT:

```
me sample
```

Within a few seconds, your screen will have been cleared of writing, the cursor will be positioned in the upper left-hand corner of the screen, and a command line will appear at the bottom of your screen.

Now type the following text. If you make a mistake, just backspace over it and retype the text. Press the carriage return or enter key after each line:

```
main()
{
	printf("Hello, world!\n");
}
```

Notice how the text appeared on the screen character by character as you typed it, much as it would appear on a piece of paper if you were using a typewriter.

Now, type **<ctrl-X><ctrl-S>**; that is, type **<ctrl-X>**, and then type **<ctrl-S>**. It does not matter whether you type capital or lower-case letters. Notice that this message has appeared at the bottom of your screen:

```
[Wrote 4 lines]
```

This command has permanently stored, or *saved*, what you typed into a file named **sample**.

Type the next few commands, which demonstrate some of the tasks that MicroEMACS can perform for you. These commands will be explained in full in the sections that follow; for now, try them to get a feel for how MicroEMACS works.

Type **<esc><**. Be sure that you type a less-than symbol '<', which on most keyboards is located just above the comma. Notice that the cursor has returned to the upper left-hand corner of the screen. Type **<esc>F**. The cursor has jumped forward by one word, and is now on the left parenthesis.

Type **<ctrl-N>**. Notice that the cursor has jumped to the next line, and is now just to the right of the left brace '{'.

Type **<ctrl-A>**. The cursor has jumped to the *beginning* of the second line of your text.

Type **<ctrl-N>** again. Now the cursor is at the beginning of the third line of the program, the **printf** statement.

Now, type **<ctrl-K>**. The third line of text has disappeared, leaving an empty space. Type **<ctrl-K>** again. The empty space where the third line of text had been has now disappeared.

Type **<esc>>**. Be sure to type a greater-than symbol '>', which on most keyboards is just above the period. The cursor has jumped to the space just below the last line of text. Now type **<ctrl-Y>**. The text that you erased a moment ago has reappeared, but in a new position on the screen.

By now, you should be feeling more at ease with typing MicroEMACS's *control* and *escape* codes. The following sections will explain what these commands mean. For now, exit from MicroEMACS by typing **<ctrl-X><ctrl-C>**, and when the message

```
Quit [y/n]?
```

appears type **y** and then **<return>**. This will return you to the shell.

Beginning a Document

This tutorial practices on file **example1.c**, **example2.c**, and **example3.c**. They are stored in the directory **/usr/src/example**. Before beginning, type the following commands to copy these files into the current directory and change their permissions:

```
cp /usr/src/sample/example?.c .
chmod +w example?.c
```

Now, type the following command to invoke MicroEMACS:

```
me example1.c
```

In a moment, the following text will appear on your screen:

```
/*
 * This is a simple C program that computes the results
 * of three different rates of inflation over the
 * span of ten years.  Use this text file to learn
 * how to use MicroEMACS commands
 * to make creating and editing text files quick,
 * efficient and easy.
 */
#include <stdio.h>
main()
{
	int i;			/* count ten years */
	float w1, w2, w3;	/* three inflated quantities */
	char  *msg = " %2d\t%f %f %f\n";/* printf string */
	i = 0;
	w1 = 1.0;
	w2 = 1.0;
	w3 = 1.0;
	for (i = 1; i<= 10; i++) {
		w1 *= 1.07;	/* apply inflation */
		w2 *= 1.08;
		w3 *= 1.10;
		printf (msg, i, w1, w2, w3);
	}
}
```

When you invoke MicroEMACS, it *copies* that file into memory. Your cursor also moved to the upper left-hand corner of the screen. At the bottom of the screen appears the *status line*, as follows:

```
-- Coherent MicroEMACS -- example1.c -- File: example1.c --
```

The word to the left, MicroEMACS, is the name of the editor. The word in the center, **example1.c**, is the name of the *buffer* that you are using. (We will describe later just what a buffer is and how you use it.) The name to the right is the name of the text file that you are editing.

Moving the Cursor

Now that you have read a text file into memory, you are ready to edit it. The first step is to learn to move the cursor.

Try these commands for yourself as we described them. That way, you will quickly acquire a feel for handling MicroEMACS's commands.

Moving the Cursor Forward

This first set of commands moves the cursor forward:

<ctrl-F>	Move forward one space
<esc>F	Move forward one word
<ctrl-E>	Move to end of line

To see how these commands work, do the following: Type the *forward* command **<ctrl-F>**. As before, it does not matter whether the letter '**F**' is upper case or lower case. The cursor has moved one space to the right, and now is over the character '*' in the first line.

Type **<esc>F**. The cursor has moved one *word* to the right, and is now over the space after the word **This**. MicroEMACS considers only alphanumeric characters when it moves from word to word. Therefore, the cursor moved from under the * to the space after the word **This**, rather than to the space after the *. Now type the *end of line* command **<ctrl-E>**. The cursor has jumped to the end of the line and is now just to the right of the **s** of the word **results**.

Moving the Cursor Backwards

The following summarizes the commands for moving the cursor backwards:

<ctrl-B>	Move back one space
<esc>B	Move back one word
<ctrl-A>	Move to beginning of line

To see how these work, first type the *backward* command **<ctrl-B>**. As you can see, the cursor has moved one space to the left, and now is over the letter **e** of the word **three**. Type **<esc>B**. The cursor has moved one *word* to the left and now is over the **t** in **three**. Type **<esc>B** again, and the cursor will be positioned on the **o** in **of**.

Type the *beginning of line* command **<ctrl-A>**. The cursor jumps to the beginnning of the line, and once again is resting over the '**/**' character in the first line.

From Line to Line

<ctrl-P>	Move to previous line
<ctrl-N>	Move to next line

These two commands move the cursor up and down the screen. Type the *next line* command **<ctrl-N>**. The cursor jumps to the space before the '*' in the next line. Type the *end of line* command **<ctrl-E>**, and the cursor moves to the end of the second line to the right of the period.

Continue to type **<ctrl-N>** until the cursor reaches the bottom of the screen. As you reached the first line in your text, the cursor jumped from its position at the right of the period on the second line to just right of the brace on the last line of the file. When you move your cursor up or down the screen, MicroEMACS tries to keep it at the same position within each line. If the line to which you are moving the cursor is not long enough to have a character at that position, MicroEMACS moves the cursor to the end of the line.

Now, practice moving the cursor back up the screen. Type the *previous line* command **<ctrl-P>**. When the cursor jumped to the previous line, it retained its position at the end of the line. MicroEMACS remembers the cursor's position on the line, and returns the cursor there when it jumps to a line long enough to have a character in that position.

Continue pressing **<ctrl-P>**. The cursor will move up the screen until it reaches the top of your text.

Repetitive Motion

Some computers repeat a command automatically if you *hold down* the control key and the character key. Try holding down **<ctrl-N>** for a moment, and see if it repeats automatically. If it does, that will speed moving your cursor around the screen, because you will not have to type the same command repeatedly.

Moving Up and Down by a Screenful of Text

The next two cursor movement commands allow you to roll forward or backwards by one screenful of text.

<ctrl-V>	Move forward one screen
<esc>V	Move back one screen

If you are editing a file with MicroEMACS that is too big to be displayed on your screen all at once, MicroEMACS displays the file in screen-sized portions (on most terminals, 22 lines at a time). The *view* commands **<ctrl-V>** and **<esc>V** allow you to roll up or down one screenful of text at a time.

Type **<ctrl-V>**. Your screen now contains only the last three lines of the file. This is because you have rolled forward by the equivalent of one screenful of text, or 22 lines.

Now, type **<esc>V**. Notice that your text rolls back onto the screen, and your cursor is positioned in the upper left-hand corner of the screen, over the character ‘**/**’ in the first line.

Moving to Beginning or End of Text

These last two cursor movement commands allow you to jump immediately to the beginning or end of your text.

<esc><	Move to beginning of text
<esc>>	Move to end of text

The *end of text* command **<esc>>** moves the cursor to the end of your text. Type **<esc>>**. Be sure to type a greater-than symbol ‘>’. Your cursor has jumped to the end of your text.

The *beginning of text* command **<esc><** will move the cursor back to the beginning of your text. Type **<esc><**. Be sure to type a less-than symbol ‘<’. The cursor has jumped back to the upper left-hand corner of your screen.

These commands move you immediately to the beginning or the end of your text, regardless of whether the text is one page or 20 pages long.

Saving Text and Quitting

If you do not wish to continue working at this time, you should *save* your text, and then *quit*.

It is good practice to save your text file every so often while you are working on it. If an accident occurs, such as a power failure, you will not lose all of your work. You can save your text with the *save* command **<ctrl-X><ctrl-S>**. Type **<ctrl-X><ctrl-S>** that is, first type **<ctrl-X>**, then type **<ctrl-S>**. If you had modified this file, the following message would appear:

```
[Wrote 23 lines]
```

The text file would have been saved to your computer’s disk. (MicroEMACS sends you messages from time to time. The messages enclosed in square brackets ‘[’ ‘]’ are for your information, and do not necessarily mean that something is wrong.) To exit from MicroEMACS, type the *quit* command **<ctrl-X><ctrl-C>**. This will return you to the shell.

Killing and Deleting

Now that you know how to move the cursor, you are ready to edit your text.

To return to MicroEMACS, type the command:

```
me example1.c
```

Within a moment, **example1.c** will be restored to your screen.

By now, you probably have noticed that MicroEMACS is always ready to insert material into your text. Unless you use the **<ctrl>** or **<esc>** keys, MicroEMACS assumes that whatever you type is text and inserts it onto your screen where your cursor is positioned.

The simplest way to erase text is simply to position the cursor to the right of the text you want to erase and backspace over it. MicroEMACS, however, also has a set of commands that allow you to erase text easily. These commands, *kill* and *delete*, behave differently; the distinction is important, and will be explained in a moment.

Deleting Vs. Killing

When MicroEMACS *deletes* text, it is erased completely and disappears forever. However, when MicroEMACS *kills* text, the text is copied into a temporary storage area in memory. This storage area is overwritten when you move the cursor and then kill additional text. Until then, however, the killed text is saved. This aspect of killing allows you to restore text that you killed accidentally, and it also allows you to move or copy portions of text from one position to another.

MicroEMACS is designed so that when it erases text, it does so beginning at the *left edge* of the cursor. This left edge is called the *current position*.

You should imagine that an invisible vertical bar separates the cursor from the character immediately to its left. As you enter the various kill and delete commands, this vertical bar moves to the right or the left with the cursor, and erases the characters it touches.

Erasing Text to the Right

The first two commands to be presented erase text to the *right*.

<ctrl-D>	Delete one character to the right
<del>	Delete one character to the right
<esc>D	Kill one word to the right

The keystrokes **<del>** and **<ctrl-D>** both delete one *character* to the right of the current position. **<esc>D** deletes one *word* to the right of the current position.

To try these commands, type **<ctrl-D>** or **<del>**. MicroEMACS erases the character '**/**' in the first line, and shifted the rest of the line one space to the left.

Now, type **<esc>D**. MicroEMACS erases the '*' character and the word **This**, and shifts the line six spaces to the left. The cursor is positioned at the *space* before the word **is**. Type **<esc>D** again. The word **is** vanishes along with the *space* that preceded it, and the line shifts *four* spaces to the left.

Remember that **<ctrl-D>** *deletes* text, but **<esc>D** *kills* text.

Erasing Text to the Left

You can erase text to the *left* with the following commands:

<backspace>	Delete one character to the left
<esc><backspace>	Kill one word to the left

To see how to erase text to the left, first type the *end of line* command **<ctrl-E>**; this will move the cursor to the right of the word **three** on the first line of text. Now, type **<backspace>**. The second **e** of the word **three** has vanished.

To erase the *word* that lies to the left of the cursor, type **<esc><backspace>**: that is, type **<esc>** and then type **<backspace>**. MicroEMACS defines a word as a string delimited by white space. For example, if you type **<esc><backspace>**, the rest of the word **three** vanishes, and the cursor moves to the white space that lies to the left of that word. If the cursor is at the beginning of a line, then this command kills the last word on the previous line of text.

Please note that erasing text with **<backspace>** *deletes* the text; that is, the text is thrown away and gone forever. Erasing text with **<esc><backspace>**, however, *kills* text; which means that the text is stored internally and can be retrieved. The distinction between *deleting* text and *killing* is described in detail below.

Erasing Lines of Text

Finally, the following command erases a line of text:

<ctrl-K>	Kill from cursor to end of line

This command kills a line of text, from the line beginning from immediately to the left of the cursor to the end of the line.

To see how this works, move the cursor to the beginning of line 2. Now, strike **<ctrl-K>**. All of line 2 has vanished and been replaced with an empty space. Strike **<ctrl-K>** again. The empty space has vanished, and the cursor is now positioned at the beginning of what used to be line 3, in the space before * **Use**.

Yanking Back (Restoring) Text

The following command allows you restore material that you have killed:

<ctrl-Y>	Yank back (restore) killed text

Remember that when you kill text, MicroEMACS temporarily stores it elsewhere. You can return this material to the screen by using the *yank back* command **<ctrl-Y>**. Type **<ctrl-Y>**. All of line 2 has returned; the cursor, however, remains at the beginning of line 3.

Quitting

When you are finished, do not save the text. If you do so, the undamaged copy of the text that you made earlier will be replaced with the present mangled copy. Rather, use the *quit* command **<ctrl-X><ctrl-C>**. Type **<ctrl-X><ctrl-C>**. On the bottom of your screen, MicroEMACS responds:

```
Quit [y/n]?
```

Reply by typing **y** and a carriage return. If you type **n**, MicroEMACS will return you to where you were in the text. MicroEMACS will now return you to the shell.

Block Killing and Moving Text

As noted above, text that is killed is stored temporarily within memory. You can yank killed text back onto your screen, and not necessarily in the spot where it was originally killed. This feature allows you to move text from one position to another.

Moving One Line of Text

You can kill and move one line of text with the following commands:

<ctrl-K>	Kill text to end of line
<ctrl-Y>	Yank back text

To test these commands, invoke MicroEMACS for the file **example1.c** by typing the following command:

```
me example1.c
```

When MicroEMACS appears, the cursor will be positioned in the upper left-hand corner of the screen.

To move the first line of text, begin by typing the *kill* command **<ctrl-K>** twice. Now, press **<esc>>** to move the cursor to the bottom of text. Finally, yank back the line by typing **<ctrl-Y>**. The line that reads

```
/* This is a simple C program that computes the results
```

is now at the bottom of your text.

Your cursor has moved to the point on your screen that is *after* the line you yanked back.

Multiple Copying of Killed Text

When text is yanked back onto your screen, it is *not* deleted from memory. Rather, it is simply copied back onto the screen. This means that killed text can be reinserted into the text more than once. To see how this is done, return to the top of the text by typing **<esc><**. Then type **<ctrl-Y>**. The line you just killed now appears as both the first and last line of the file.

The killed text will not be erased from its temporary storage until you move the cursor and then kill additional text. If you kill several lines or portions of lines in a row, all of the killed text will be stored in the buffer; if you are not careful, you may yank back a jumble of accumulated text.

Kill and Move a Block of Text

If you wish to kill and move more than one line of text at a time, use the following commands:

<ctrl-@>	Set mark
<esc>.	Set mark
<ctrl-W>	Kill block of text
<ctrl-Y>	Yank back text

If you wish to kill a block of text, you can either type the *kill* command **<ctrl-K>** repeatedly to kill the block one line at a time, or you can use the *block kill* command **<ctrl-W>**. To use this command, you must first set a *mark* on the screen, an invisible character that acts as a signal to the computer. The mark can be set with either **<esc>.** or **<ctrl-@>**.

Once the mark is set, you must move your cursor to the other end of the block of text you wish to kill, and then strike **<ctrl-W>**. The block of text will be erased, and will be ready to be yanked back elsewhere.

Try this out on **example1.c**. Type **<esc><** to move the cursor to the upper left-hand corner of the screen. Then type the *set mark* command **<ctrl-@>**. MicroEMACS will respond with the message

```
[Mark set]
```

at the bottom of your screen. Now, move the cursor down six lines, and type **<ctrl-W>**. Note how the block of text you marked out has disappeared.

Move the cursor to the bottom of your text. Type **<ctrl-Y>**. The killed block of text has now been reinserted.

When you yank back text, be sure to position the cursor at the *exact* point where you want the text to be yanked back. This will ensure that the text will be yanked back in the proper place. To try this out, move your cursor up six lines. Be careful that the cursor is at the *beginning* of the line. Now, type **<ctrl-Y>** again. The text reappeared *above* where the cursor was positioned, and the cursor has not moved from its position at the beginning of the line which is not what would have happened had you positioned it in the middle or at the end of a line.

Although the text you are working with has only 23 lines, you can move much larger portions of text using only these three commands. Remember, too, that you can use this technique to duplicate large portions of text at several positions to save yourself considerable time in typing and reduce the number of possible typographical errors.

Capitalization and Other Tools

The next commands perform a number of tasks to help with your editing. Before you begin this section, destroy the old text on your screen with the *quit* command **<ctrl-X><ctrl-C>**, and read into MicroEMACS a fresh copy of the program, as you did earlier.

Capitalization and Lowercasing

The following MicroEMACS commands automatically capitalize a word (that is, make the first letter of a word upper case), or make an entire word upper case or lower case.

<esc>C	Capitalize a word
<esc>L	Lowercase from cursor to end of word
<esc>U	Uppercase from cursor to end of word

To try these commands, do the following: First, move the cursor to the letter **d** of the word *different* on line 2. Type the *capitalize* command **<esc>C**. The word is now capitalized, and the cursor is now positioned at the space after the word. Move the cursor forward so that it is over the letter **t** in **rates**. Press **<esc>C** again. The word changes to **raTes**. When you press **<esc>C**, MicroEMACS capitalizes the *first* letter the cursor meets.

MicroEMACS can also change a word to all upper case or all lower case. (There is very little need for a command that will change only the first character of an upper-case word to lower case, so it is not included.)

Type **<esc>B** to move the cursor so that it is again to the left of the word **Different**. It does not matter if the cursor is directly over the **D** or at the space to its left; as you will see, this means that you can capitalize or lowercase a number of words in a row without having to move the cursor.

Type the *uppercase* command **<esc>U**. The word is now spelled **DIFFERENT**, and the cursor has jumped to the space after the word.

Again, move the cursor to the left of the word **DIFFERENT**. Type the *lowercase* command **<esc>L**. The word has changed back to **different**. Now, move the cursor to the space at the beginning of line 3 by typing **<ctrl-N>** then **<ctrl-A>**. Type **<esc>L** once again. The character “*’ is not affected by the command, but the letter **U** is now lower case. **<esc>L** not only shifts a word that is all upper case to lower case: it can also un-capitalize a word.

The *uppercase* and *lowercase* commands stop at the first punctuation mark they meet *after* the first letter they find. This means that, for example, to change the case of a word with an apostrophe in it you must type the appropriate command twice.

Transpose Characters

MicroEMACS allows you to reverse the position of two characters, or *transpose* them, with the *transpose* command **<ctrl-T>**.

Type **<ctrl-T>**. MicroEMACS transposes the character that is under the cursor with the character immediately to its *left*. In this example,

```
* use this
```

in line 3 now appears:

```
* us ethis
```

The space and the letter **e** have been transposed. Type **<ctrl-T>** again. The characters have returned to their original order.

Screen Redraw

<ctrl-L>	Redraw screen

Occasionally, while you are working on a text another COHERENT user will write or mail you a message. COHERENT will write the message directly on your screen, which scrambles your screen. A message sent from another user or a message from the COHERENT system is *not* recorded into your text; however, you may wish to erase the message and continue editing. The *redraw screen* command **<ctrl-L>** will redraw your screen to the way it was before it was scrambled.

Type **<ctrl-L>**. Notice how the screen flickers and the text is rewritten. Had your screen been spoiled by extraneous material, that material would have been erased and the original text rewritten.

The **<ctrl-L>** command also has another use: it can move the line on which the cursor is positioned to the center of the screen. If you have a file that contains more than one screenful of text and you wish to have that particular line in the center of the screen, position the cursor on that line and type **<ctrl-U><ctrl-L>**. You will see a prompt that says

```
Arg: 4
```

The meaning of this prompt is explained below; for now, ignore it and press **<return>**. Immediately, MicroEMACS redraws the screen and places the line you selected in the center of the screen.

Return Indent

<ctrl-J>	Return and indent

You may often be faced with a situation in which, for the sake of programming style, you need to indent many lines of text: before every line you must tab the correct number of times before typing the text. These *block indents* can be a time-consuming typing chore. The MicroEMACS **<ctrl-J>** command makes this task easier. **<ctrl-J>** moves the cursor to the next line on the screen and automatically positions the cursor at the previous line's level of indentation.

To see how this works, first move the cursor to the line that reads

```
w3 *= 1.10:
```

Press **<ctrl-E>**, to move the cursor to the end of the line. Now, type **<ctrl-J>**.

As you can see, a new line opens up and the cursor is indented the same amount as the previous line. Type

```
/* Here is an example of auto-indentation */
```

This line of text begins directly under the previous line.

Word Wrap

<ctrl-X>F Set word wrap

Although you have not yet had much opportunity to use it, MicroEMACS will automatically wrap text that you are typing. Word-wrapping is controlled with the *word wrap* command **<ctrl-X>F**. To see how the word wrap command works, first exit from MicroEMACS by typing **<ctrl-X><ctrl-C>**; then reinvoke MicroEMACS by typing

```
me cucumber
```

When MicroEMACS re-appears, type the following text; however, do *not* type any carriage returns:

```
A cucumber should be
well sliced, and dressed
with pepper and vinegar,
and then thrown out, as
good for nothing.
```

When you reached the edge of your screen, a dollar sign was printed and you were allowed to continue typing. MicroEMACS accepted the characters you typed, but it placed them at a location beyond the right edge of your screen.

Now, move to the beginning of the next line and type **<ctrl-U>**. MicroEMACS will reply with the message:

```
Arg: 4
```

Type **30**. The line at the bottom of your screen now appears as follows:

```
Arg: 30
```

(The use of the *argument* command **<ctrl-U>** will be explained in a few minutes.) Now type the *word-wrap* command **<ctrl-X>F**. MicroEMACS will now say at the bottom of your screen:

```
[Wrap at column 30]
```

This sequence of commands has set the word-wrap function, and told it to wrap to the next line all words that extend beyond the 30th column on your screen.

The *word wrap* feature automatically moves your cursor to the beginning of the next line once you type past a preset border on your screen. When you first enter MicroEMACS, that limit is automatically set at the first column, which in effect means that word wrap has been turned off.

When you type prose for a report or a letter of some sort, you probably will want to set the border at the 65th column, so that the printed text will fit neatly onto a sheet of paper. If you are using MicroEMACS to type in a program, however, you probably will want to leave word wrap off, so you do not accidentally introduce carriage returns into your code.

To test word wrapping, type the above text again, without using the carriage return key. When you finish, it should appear as follows:

```
A cucumber should be well
sliced, and dressed with
pepper and vinegar, and then
thrown out, as good for
nothing.
```

MicroEMACS automatically moved your cursor to the next line when you typed a space character after the 30th column on your screen.

If you wish to fix the border at some special point on your screen but do not wish to go through the tedium of figuring out how many columns from the left it is, simply position the cursor where you want the border to be, type **<ctrl-X>F**, and then type a carriage return. When **<ctrl-X>F** is typed without being preceded by a **<ctrl-U>** command, it sets the word-wrap border at the point your cursor happens to be positioned. When you do this, MicroEMACS will then print a message at the bottom of your terminal that tells you where the word-wrap border is now set.

To re-word wrap the text between the cursor and the mark, type **<ctrl-X>H**.

If you wish to turn off the word wrap feature again, simply set the word wrap border to one.

Search and Reverse Search

When you edit a large text, you may wish to change particular words or phrases. To do this, you can roll through the text and read each line to find them; or you can have MicroEMACS find them for you. Before you continue, close the present file by typing **<ctrl-X> <ctrl-C>**; then reinvoke the editor to edit the file **example1.c**, as you did before. The following sections perform some exercises with this file.

Search Forward

<ctrl-S>	Search forward incrementally
<esc>S	Search forward with prompt

As you can see from the display, MicroEMACS has two ways to search forward: incrementally, and with a prompt.

An *incremental* search is one in which the search is performed as you type the characters. To see how this works, first type the *beginning of text* command **<esc><** to move the cursor to the upper left-hand corner of your screen. Now, type the *incremental search* command **<ctrl-S>**. MicroEMACS will respond by prompting with the message

```
i-search forward []
```

at the bottom of the screen.

We will now search for the pointer ***msg**. Type the letters ***msg** one at a time, starting with *. The cursor has jumped to the first place that a * was found: at the second character of the first line. The cursor moves forward in the text file and the message at the bottom of the screen changes to reflect what you have typed.

Now type **m**. The cursor has jumped ahead to the letter **s** in ***msg**. Type **s**. The cursor has jumped ahead to the letter **g** in ***msg**. Finally, type **g**. The cursor is over the space after the token ***msg**. Finally, type **<esc>** to end the string. MicroEMACS replies with the message

```
[Done]
```

which indicates that the search is finished.

If you attempt an incremental search for a word that is not in the file, MicroEMACS will find as many of the letters as it can, and then give you an error message. For example, if you tried to search incrementally for the word ***msgs**, MicroEMACS would move the cursor to the phrase ***msg**; when you typed ‘s’, it would tell you

```
failing i-search forward: *msgs
```

With the *prompt search*, however, you type in the word all at once. To see how this works, type **<esc><**, to return to the top of the file. Now, type the *prompt search* command **<esc>S**. MicroEMACS responds by prompting with the message

```
Search [*msgs]:
```

at the bottom of the screen. The word ***msgs** is shown because that was the last word for which you searched, and so it is kept in the search buffer.

Type in the words **editing text**, then press the carriage return. Notice that the cursor has jumped to the period after the word **text** in the next to last line of your text. MicroEMACS searched for the words **editing text**, found them, and moved the cursor to them.

If the word you were searching for was not in your text, or at least was not in the portion that lies between your cursor and the end of the text, MicroEMACS would not have moved the cursor, and would have displayed the message

```
Not found
```

at the bottom of your screen.

Reverse Search

<ctrl-R>	Search backwards incrementally
<esc>R	Search backwards with prompt

The search commands, useful as they are, can only search forward through your text. To search backwards, use the reverse search commands **<ctrl-R>** and **<esc>R**. These work exactly the same as their forward-searching counterparts, except that they search toward the beginning of the file rather than toward the end.

For example, type **<esc>R**. MicroEMACS replies with the message

```
Reverse search [editing text]:
```

at the bottom of your screen. The words in square brackets are the words you entered earlier for the *search* command; MicroEMACS remembered them. If you wanted to search for **editing text** again, you would just press the carriage return. For now, however, type the word **program** and press the carriage return.

Notice that the cursor has jumped so that it is under the letter **p** of the word **program** in line 1. When you search forward, the cursor moves to the *space after* the word for which you are searching, whereas when you reverse search the cursor moves to the *first letter* of the word for which you are searching.

Cancel a Command

<ctrl-G> Cancel a search command

As you have noticed, the commands to move the cursor or to delete or kill text all execute immediately. Although this speeds your editing, it also means that if you type a command by mistake, it executes before you can stop it.

The *search* and *reverse search* commands, however, wait for you to respond to their prompts before they execute. If you type **<esc>S** or **<esc>R** by accident, MicroEMACS will interrupt your editing and wait for you to initate a search that you do not want to perform. You can evade this problem, however, with the *cancel* command **<ctrl-G>**. This command tells MicroEMACS to ignore the previous command.

To see how this command works, type **<esc>R**. When the prompt appears at the bottom of your screen, type **<ctrl-G>**. Three things happen: your terminal beeps, the characters **^G** appear at the bottom of your screen, and the cursor returns to where it was before you first typed **<esc>R**. The **<esc>R** command has been cancelled, and you are free to continue editing.

If you cancel an *incremental search* command, **<ctrl-S>** or **<esc-S>**, the cursor returns to where it was before you began the search. For example, type **<esc><** to return to the top of the file. Now type **<ctrl-S>** to begin an incremental search, and type **m**. When the cursor moves to the **m** in **simple**, type **<ctrl-G>**. Your cursor returns to the top of the file, where you began the search.

Search and Replace

<esc>% Search and replace

MicroEMACS also gives you a powerful function that allows you to search for a string and replace it with a keystroke. You can do this by executing the *search and replace* command **<esc>%**.

To see how this works, move to the top of the text file by typing **<esc><**; then type **<esc>%**. You will see the following message at the bottom of your screen:

```
Old string [m]
```

As an exercise, type **msg**, and then press **<return>**. MicroEMACS will then ask:

```
New string:
```

Type **message**, and press the carriage return. As you can see, the cursor jumps to the first occurrence of the string **msg**, and prints the following message at the bottom of your screen:

```
Query replace: [msg] -> [message]
```

MicroEMACS is asking if it should proceed with the replacement. Type a carriage return: this displays the options that are available to you at the bottom of your screen:

```
<SP>[,] replace, [.] rep-end, [n] dont, [!] repl rest <C-G> quit
```

The options are as follows:

Typing a space or a comma executes the replacement, and moves the cursor to the next occurrence of the old string; in this case, it replaces **msg** with **message**, and moves the cursor to the next occurrence of **msg**.

Typing a period '.' replaces this one occurrence of the old string and ends the search and replace procedure. In this example, typing a period replaces this one occurrence of **msg** with **message** and ends the procedure.

Typing the letter ‘n’ tells MicroEMACS *not* to replace this instance of the old string, but move to the next occurrence of the old string. In this case, typing ‘n’ does *not* replace **msg** with **message**, and the cursor jumps to the next place where **msg** occurs.

Typing an exclamation point ‘!’ tells MicroEMACS to replace all instances of the old string with the new string automatically, without checking with you any further. In this example, typing ‘!’ replaces all instances of **msg** with **message** without further queries from MicroEMACS. When you finish searching and replacing, MicroEMACS displays a message that tells how many replacements it made.

Typing **<ctrl-G>** aborts the search and replace procedure.

Saving Text and Exiting

This set of basic editing commands allows you to save your text and exit from the MicroEMACS program. They are as follows:

<ctrl-X><ctrl-S>	Save text
<ctrl-X><ctrl-W>	Write text to a new file
<ctrl-Z>	Save text and exit
<ctrl-X><ctrl-C>	Exit without saving text

You have used two of these commands already: the *save* command **<ctrl-X><ctrl-S>** and the *quit* command **<ctrl-X><ctrl-C>**, which respectively allow you to save text or to exit from MicroEMACS without saving text. (Commands that begin with **<ctrl-X>** are called *extended* commands; they are used frequently in the commands described later in this tutorial.)

Write Text to a New File

<ctrl-X> <ctrl-W>	Write text to a new file

If you wish, you can copy the text you are currently editing to a text file other than the one from which you originally read the text. Do this with the *write* command **<ctrl-X><ctrl-W>**.

To test this command, type **<ctrl-X><ctrl-W>**. MicroEMACS displays the following message on the bottom of your screen:

```
Write file:
```

MicroEMACS is asking for the name of the file into which you wish to write the text. Type **sample**. MicroEMACS replies:

```
[Wrote 23 lines]
```

The 23 lines of your text have been copied to a new file called **sample**. The status line at the bottom of your screen has changed to read as follows:

```
-- Coherent MicroEMACS -- example1.c -- File: sample --------------
```

The significance of the change in file name will be discussed in the second half of this tutorial.

Before you copy text into a new file, be sure that you have not selected a file name that is already being used. If you do, MicroEMACS will erase whatever is stored under that file name, and the text created with MicroEMACS will be stored in its place.

Save Text and Exit

Finally, the *store* command **<ctrl-Z>** will save your text *and* move you out of the MicroEMACS editor. To see how this works, watch the bottom line of your terminal carefully and type **<ctrl-Z>**. MicroEMACS has saved your text, and now you can issue commands directly to the shell.

Advanced Editing

The second half of this tutorial introduces the advanced features of MicroEMACS.

The techniques described here will help you execute complex editing tasks with minimal trouble. You will be able to edit more than one text at a time, display more than one file on your screen at a time, enter a long or complicated phrase repeatedly with only one keystroke, and give commands to COHERENT without having to exit from MicroEMACS.

Before beginning, however, you must prepare a new text file. Type the following command:

```
me example2.c
```

In a moment, **example2.c** will appear on your screen, as follows:

```
/* Use this program to get better acquainted
 * with the MicroEMACS interactive screen editor.
 * You can use this text to learn some of the
 * more advanced editing features of MicroEMACS.
 */

#include <stdio.h>
main()
{
        FILE *fp;
        int ch;
        int filename[20];

        printf("Enter file name: ");
        gets(filename);

        if ((fp =fopen(filename,"r")) !=NULL) {
                while ((ch = fgetc(fp)) != EOF)
                        fputc(ch, stdout);
        } else
                printf("Cannot open %s.\n", filename);
        fclose(fp);
}
```

Arguments

Most of the commands already described in this tutorial can be used with *arguments*. An argument is a subcommand that tells MicroEMACS to execute a command a given number of times. With MicroEMACS, arguments are introduced by typing **<ctrl-U>**.

Arguments: Default Values

By itself, **<ctrl-U>** sets the argument at *four*. To illustrate this, first type the *next line* command **<ctrl-N>**. By itself, this command moves the cursor down one line, from being over the '**/**' at the beginning of line 1, to being over the *space* at the beginning of line 2.

Now, type **<ctrl-U>**. MicroEMACS replies with the message:

```
Arg: 4
```

Now type **<ctrl-N>**. The cursor jumps down *four* lines, from the beginning of line 2 to the letter **m** of the word **main** at the beginning of line 6.

Type **<ctrl-U>**. The line at the bottom of the screen again shows that the value of the argument is four. Type **<ctrl-U>** again. Now the line at the bottom of the screen reads:

```
Arg: 16
```

Type **<ctrl-U>** once more. The line at the bottom of the screen now reads:

```
Arg: 64
```

Each time you type **<ctrl-U>**, the value of the argument is *multiplied* by four. Type the *forward* command **<ctrl-F>**. The cursor has jumped ahead 64 characters, and is now over the **i** of the word **file** in the *printf* statement in line 11.

Selecting Values

Naturally, an argument does not have to be a power of four. You can set the argument to whatever number you wish, simply by typing **<ctrl-U>** and then typing the number you want.

For example, type **<ctrl-U>**, and then type **3**. The line at the bottom of the screen now reads:

```
Arg: 3
```

Type the *delete* command **<esc>D**. MicroEMACS has deleted three words to the right.

You can use arguments to increase the power of any *cursor movement* command, or any *kill* or *delete* command. The sole exception is **<ctrl-W>**, the *block kill* command.

Deleting With Arguments: An Exception

Killing and *deleting* were described in the first part of this tutorial. They were said to differ in that text that was killed was stored in a special area of the computer and could be yanked back, whereas text that was deleted was erased outright. However, there is one exception to this rule: any text that is deleted using an argument can also be yanked back.

To see how this works, first type the *begin text* command **<esc><** to move the cursor to the upper left-hand corner of the screen. Then, type **<ctrl-U> 5 <ctrl-D>**. The word **Use** has disappeared. Move the cursor to the right until it is between the words **better** and **acquainted**, then type **<ctrl-Y>**. The word **Use** has been moved within the line (although the spaces around it have not been moved). This function is very handy, and should greatly speed your editing.

Remember, too, that unless you move the cursor between one set of deletions and another, the computer's storage area will not be erased, and you may yank back a jumble of text.

Buffers and Files

Before beginning this section, replace the edited copy of the text on your screen with a fresh copy. Type the *quit* command **<ctrl-X><ctrl-C>** to exit from MicroEMACS without saving the text; then return to MicroEMACS to edit the file **example2.c**, as you did earlier.

Now, look at the status line at the bottom of your screen. It should appear as follows:

```
-- Coherent MicroEMACS -- example2.c -- File: example2.c -------------
```

As noted in the first half of this tutorial, the name on the left of the command line is that of the program. The name in the middle is the name of the *buffer* with which you are now working, and the name to the right is the name of the *file* from which you read the text.

Definitions

A *file* is a mass of text that has been given a name and has been permanently stored on your disk. A *buffer* is a portion of the computer's memory that has been set aside for you to use, which may be given a name, and into which you can put text temporarily. You can place text into the buffer either by typing it at your keyboard or by *copying* it from a file.

Unlike a file, a buffer is not permanent: if your computer were to stop working (because you turned the power off, for example), a file would not be affected, but a buffer would be erased.

You must *name* your files because you work with many different files, and you must have some way to tell them apart. Likewise, MicroEMACS allows you to *name* your buffers, because MicroEMACS allows you to work with more than one buffer at a time.

File and Buffer Commands

MicroEMACS gives you a number of commands for handling files and buffers. These include the following:

<ctrl-X><ctrl-W>	Write text to file
<ctrl-X><ctrl-F>	Rename file
<ctrl-X><ctrl-R>	Replace buffer with named file
<ctrl-X><ctrl-V>	Switch buffer or create a new buffer
<ctrl-X>K	Delete a buffer
<ctrl-X><ctrl-B>	Display the status of each buffer

Write and Rename Commands

The *write* command **<ctrl-X><ctrl-W>** was introduced earlier when the commands for saving text and exiting were discussed. To review, **<ctrl-X><ctrl-W>** changes the name of the file into which the text is saved, and then copies the text into that file.

Type **<ctrl-X><ctrl-W>**. MicroEMACS responds by printing

```
Write file:
```

on the last line of your screen.

Type **junkfile**, then **<return>**. Two things happen: First, MicroEMACS writes the message

```
[Wrote 21 lines]
```

at the bottom of your screen. Second, the name of the file shown on the status line changes from **example2.c** to **junkfile**. MicroEMACS is reminding you that your text is now being saved into the file **junkfile**.

The *file rename* command **<ctrl-X><ctrl-F>** allows you rename the file to which you are saving text, *without* automatically writing the text to it. Type **<ctrl-X><ctrl-F>**. MicroEMACS will reply with the prompt:

```
Name:
```

Type **example2.c** and **<return>**. MicroEMACS does *not* send you a message that lines were written to the file; however, the name of the file shown on the status line has changed from **junkfile** back to **example2.c**.

Replace Text in a Buffer

The *replace* command **<ctrl-X><ctrl-R>** allows you to replace the text in your buffer with the text from another file.

Suppose, for example, that you had edited **example2.c** and saved it, and now wished to edit **example1.c**. You could exit from MicroEMACS, then re-invoke MicroEMACS for the file **example1.c**, but this is cumbersome. A more efficient way is to simply replace the **example2.c** in your buffer with **example1.c**.

Type **<ctrl-X><ctrl-R>**. MicroEMACS replies with the prompt:

```
Read file:
```

Type **example1.c**. Notice that **example2.c** has rolled away and been replaced with **example1.c**. Now, check the status line. Notice that although the name of the *buffer* is still **example2.c**, the name of the *file* has changed to **example1.c**. You can now edit **example1.c**; when you save the edited text, MicroEMACS will copy it back into the file **example1.c** unless, of course, you again choose to rename the file.

Visiting Another Buffer

The last command of this set, the *visit* command **<ctrl-X><ctrl-V>**, allows you to create more than one buffer at a time, to jump from one buffer to another, and move text between buffers. This powerful command has numerous features.

Before beginning, however, straighten up your buffer by replacing **example1.c** with **example2.c**. Type the *replace* command **<ctrl-X><ctrl-R>**; when MicroEMACS replies by asking

```
Read file:
```

at the bottom of your screen, type **example2.c**.

You should now have the file **example2.c** read into the buffer named **example2.c**.

Now, type the *visit* command **<ctrl-X><ctrl-V>**. MicroEMACS replies with the prompt

```
Visit file:
```

at the bottom of the screen. Now type **example1.c**. Several things happen. **example2.c** rolls off the screen and is replaced with **example1.c**; the status line changes to show that both the buffer name and the file name are now **example1.c**; and the message

```
[Read 23 lines]
```

appears at the bottom of the screen.

This does *not* mean that your previous buffer has been erased, as it would have been had you used the *replace* command **<ctrl-X><ctrl-R>**. MicroEMACS is still keeping **example2.c** "alive" in a buffer and it is available for editing; however, it is not being shown on your screen at the present moment.

Type **<ctrl-X><ctrl-V>** again, and when the prompt appears, type **example2.c**. **example1.c** scrolls off your screen and is replaced by **example2.c**, and the message

```
[Old buffer]
```

appears at the bottom of your screen. You have just jumped from one buffer to another.

Move Text From One Buffer to Another

The *visit* command **<ctrl-X><ctrl-V>** not only allows you to jump from one buffer to another: it allows you to *move text* from one buffer to another as well. The following example shows how you can do this.

First, kill the first line of **example2.c** by typing the *kill* command **<ctrl-K>** twice. This removes both the line of text *and* the space that it occupied. If you did not remove the space as well the line itself, no new line would be created for the text when you yank it back. Next, type **<ctrl-X><ctrl-V>**. When the prompt

```
Visit file:
```

appears at the bottom of your screen, type **example1.c**. When **example1.c** has rolled onto your screen, type the *yank back* command **<ctrl-Y>**. The line you killed in **example2.c** has now been moved into **example1.c**.

Checking Buffer Status

The number of buffers you can use at any one time is limited only by the size of your computer. You should create only as many buffers as you need to use immediately; this will help the computer run efficiently.

To help you keep track of your buffers, MicroEMACS has the *buffer status* command **<ctrl-X><ctrl-B>**. Type **<ctrl-X><ctrl-B>**. The status line moves up to the middle of the screen, and the bottom half of your screen is replaced with the following display:

```
C    Size Lines   Buffer         File
-    ---- -----   ------         ----
*     655    24   example1.c     example1.c
*     403    20   example2.c     example2.c
```

This display is called the *buffer status window*. The use of windows will be discussed more fully in the following section.

The letter **C** over the leftmost column stands for **Changed**. An asterisk indicates that that buffer has been changed since it was last saved, whereas a space means that the buffer has not been changed. **Size** indicates the buffer's size, in number of characters; **Buffer** lists the buffer name, and **File** lists the file name.

Now, kill the second line of **example1.c** by typing the *kill* command **<ctrl-K>**. Then type **<ctrl-X><ctrl-B>** once again. The size of the buffer **example1.c** shrinks from 657 characters to 595 to reflect the decrease in the size of the buffer.

To make this display disappear, type the *one window* command **<ctrl-X>1**. This command will be discussed in full in the next section.

Renaming a Buffer

One more point must be covered with the *visit* command. COHERENT does not allow you to have more than one file with the same name. For the same reason, MicroEMACS does not allow you to have more than one *buffer* with the same name.

Ordinarily, when you visit a file that is not already in a buffer, MicroEMACS creates a new buffer and gives it the same name as the file you are visiting. However, if for some reason you already have a buffer with the same name as the file you wish to visit, MicroEMACS stops and asks you to give a new, different name to the buffer it is creating.

For example, suppose that you wanted to visit a new *file* named **sample**, but you already had a *buffer* named **sample**. MicroEMACS would stop and give you this prompt at the bottom of the screen:

```
Buffer name:
```

You would type in a name for this new buffer. This name could not duplicate the name of any existing buffer. MicroEMACS would then read the file **sample** into the newly named buffer.

Delete a Buffer

If you wish to delete a buffer, simply type the *delete buffer* command **<ctrl-X>K**. This command allows you to delete only a buffer that is hidden, not one that is being displayed.

Type **<ctrl-X>K**. MicroEMACS will give you the prompt:

```
Kill buffer:
```

Type **example2.c**. Because you have changed the buffer, MicroEMACS asks:

```
Discard changes [y/n]?
```

Type **y**, then **<return>**. Now, type the *buffer status* command **<ctrl-X><ctrl-B>**. The buffer status window no longer shows the buffer **example2.c**. Although the prompt refers to *killing* a buffer, the buffer is in fact *deleted* and cannot be yanked back.

Windows

Before beginning this section, it will be necessary to create a new text file. Exit from MicroEMACS by typing the *quit* command **<ctrl-X><ctrl-C>**; then reinvoke MicroEMACS for the text file **example1.c** as you did earlier.

Now, copy **example2.c** into a buffer by typing the **visit** command **<ctrl-X><ctrl-V>**. When the message

```
Visit file:
```

appears at the bottom of your screen, type **example2.c**. MicroEMACS reads **example2.c** into a buffer, and shows the message

```
[Read 21 lines]
```

at the bottom of your screen.

Finally, copy a new text, called **example3.c**, into a buffer. (You copied this file into the current directory when you began this tutorial.) Type **<ctrl-X><ctrl-V>** again. When MicroEMACS asks which file to visit, type **example3.c**. The message

```
[Read 123 lines]
```

appears at the bottom of your screen.

The first screenful of text appears as follows:

```
/*
 * Factor prints out the prime factorization of numbers.
 * If there are any arguments, then it factors these.  If
 * there are no arguments, then it reads stdin until
 * either EOF or the number zero or a non-numeric
 * non-white-space character.  Since factor does all of
 * its calculations in double format, the largest number
 * which can be handled is quite large.
 */
#include <stdio.h>
#include <math.h>
#include <ctype.h>

#define NUL '\0'
#define ERROR 0x10 /* largest input base */
#define MAXNUM 200 /* max number of chars in number */

main(argc, argv)
int argc;
register char *argv[];

-- Coherent MicroEMACS -- example3.c -- File: example3.c --------------
```

At this point, **example3.c** is on your screen, and **example1.c** and **example2.c** are hidden.

You could edit first one text and then another, while remembering just how things stood with the texts that were hidden; but it would be much easier if you could display all three texts on your screen simultaneously. MicroEMACS allows you to do just that by using *windows*.

Creating Windows and Moving Between Them

A *window* is a portion of your screen that can be manipulated independent of the rest of the screen. The following commands let you create windows and move between them:

<ctrl-X>2	Create a window
<ctrl-X>1	Delete extra windows
<ctrl-X>N	Move to next window
<ctrl-X>P	Move to previous window

The best way to grasp how a window works is to create one and work with it. To begin, type the *create a window* command **<ctrl-X>2**.

Your screen is now divided into two parts, an upper and a lower. The same text is in each part, and the command lines give **example3.c** for the buffer and file names. Also, note that you still have only one cursor, which is in the upper left-hand corner of the screen.

The next step is to move from one window to another. Type the *next window* command **<ctrl-X>N**. Your cursor has now jumped to the upper left-hand corner of the *lower* window.

Type the *previous window* command **<ctrl-X>P**. Your cursor has returned to the upper left-hand corner of the top window.

Now, type **<ctrl-X>2** again. The window on the top of your screen is now divided into two windows, for a total of three on your screen. Type **<ctrl-X>2** again. The window at the top of your screen has again divided into two windows, for a total of four.

It is possible to have as many as 11 windows on your screen at one time, although each window will show only the control line and one or two lines of text. Neither **<ctrl-X>2** nor **<ctrl-X>1** can be used with arguments.

Now, type the *one window* command **<ctrl-X>1**. All of the extra windows have been eliminated, or *closed*.

Enlarging and Shrinking Windows

When MicroEMACS creates a window, it divides into half the window in which the cursor is positioned. You do not have to leave the windows at the size MicroEMACS creates them, however. If you wish, you may adjust the relative size of each window on your screen, using the *enlarge window* and *shrink window* commands:

<ctrl-X>Z	Enlarge window
<ctrl-X><ctrl-Z>	Shrink window

To see how these work, first type **<ctrl-X>2** twice. Your screen is now divided into three windows: two in the top half of your screen, and the third in the bottom half.

Now, type the *enlarge window* command **<ctrl-X>Z**. The window at the top of your screen is now one line bigger: it has borrowed a line from the window below it. Type **<ctrl-X>Z** again. Once again, the top window has borrowed a line from the middle window.

Now, type the *next window* command **<ctrl-X>N** to move your cursor into the middle window. Again, type the *enlarge window* command **<ctrl-X>Z**. The middle window has borrowed a line from the bottom window, and is now one line larger.

The *enlarge window* command **<ctrl-X>Z** allows you to enlarge the window your cursor is in by borrowing lines from another window, provided that you do not shrink that other window out of existence. Every window must have at least two lines in it: one command line and one line of text.

The *shrink window* command **<ctrl-X><ctrl-Z>** allows you to decrease the size of a window. Type **<ctrl-X><ctrl-Z>**. The present window is now one line smaller, and the lower window is one line larger because the line borrowed earlier has been returned.

The *enlarge window* and *shrink window* commands can also be used with arguments introduced with **<ctrl-U>**. However, remember that MicroEMACS will not accept an argument that would shrink another window out of existence.

Displaying Text Within a Window

Displaying text within the limited area of a window can present special problems. The *view* commands **<ctrl-V>** and **<esc>V** roll window-sized portions of text up or down, but you may become disoriented when a window shows only four or five lines of text at a time. Therefore, three special commands are available for displaying text within a window:

<ctrl-X><ctrl-N>	Scroll down
<ctrl-X><ctrl-P>	Scroll up
<esc>!	Move within window

Two commands allow you to move your text by one line at a time, or *scroll* it: the *scroll up* command **<ctrl-X><ctrl-N>**, and the *scroll down* command **<ctrl-X><ctrl-P>**.

Type **<ctrl-X><ctrl-N>**. The line at the top of your window has vanished, a new line has appeared at the bottom of your window, and the cursor is now at the beginning of what had been the second line of your window.

Now type **<ctrl-X><ctrl-P>**. The line at the top that had vanished earlier has now returned, the cursor is at the beginning of it, and the line at the bottom of the window has vanished. These commands allow you to move forward in your text slowly so that you do not become disoriented.

Both of these commands can be used with arguments introduced by **<ctrl-U>**.

The third special movement command is the *move within window* command **<esc>!**. This command moves the line your cursor is on to the top of the window.

To try this out, move the cursor down three lines by typing **<ctrl-U>3<ctrl-N>**; now type **<esc>!**. (Be sure to type an exclamation point '!', not a numeral one '1', or your window will vanish. The command **<esc>1** is explained below.) The line to which you had moved the cursor is now the first line in the window, and three new lines have scrolled up from the bottom of the window. You will find this command to be very useful as you become more experienced at using windows.

All three special movement commands can also be used when your screen has no extra windows, although you will not need them as much.

One Buffer

Now that you have been introduced to the commands for manipulating windows, you can begin to use windows to speed your editing.

To begin with, scroll up the window you are in until you reach the top line of your text. You can do this either by typing the *scroll up* command **<ctrl-X><ctrl-P>** several times, or by typing **<esc><**.

Kill the first line of text with the *kill* command **<ctrl-K>**. The first line of text has vanished from all three windows. Now, type **<ctrl-Y>** to yank back the text you just killed. The line has reappeared in all three windows.

The main advantage to displaying one buffer with more than one window is that each window can display a different portion of the text. This can be quite helpful if you are editing or moving a large text.

To demonstrate this, do the following: First, move to the end of the text in your present window by typing the *end of text* command **<esc>>**, then typing the *previous line* command **<ctrl-P>** four times. Now, kill the last four lines.

You could move the killed lines to the beginning of your text by typing the *beginning of text* command **<esc><**; however, it is more convenient simply to type the *next window* command **<ctrl-X>N**, which moves you to the beginning of the text as displayed in the next window. MicroEMACS remembers a different cursor position for each window.

Now yank back the four killed lines by typing **<ctrl-Y>**. You can simultaneously observe that the lines have been removed from the end of your text and that they have been restored at the beginning.

Multiple Buffers

Windows are especially helpful when they display more than one text. Remember that at present you are working with *three* buffers, named **example1.c**, **example2.c**, and **example3.c**, although your screen is displaying only the text **example3.c**. To display a different text in a window, use the *switch buffer* command **<ctrl-X>B**.

Type **<ctrl-X>B**. When MicroEMACS asks

```
Use buffer:
```

at the bottom of the screen, type **example1.c**. The text in your present window is replaced with **example1.c**. The command line in that window changes, too, to reflect the fact that the buffer and the file names are now **example1.c**.

Moving and Copying Text Among Buffers

It is now very easy to copy text among buffers. To see how this is done, first kill the first line of **example1.c** by typing the **<ctrl-K>** command twice. Yank back the line immediately by typing **<ctrl-Y>**. Remember, the line you killed has *not* been erased from its special storage area, and may be yanked back any number of times.

Now, move to the previous window by typing **<ctrl-X>P**, then yank back the killed line by typing **<ctrl-Y>**. This technique can also be used with the *block kill* command **<ctrl-W>** to move large amounts of text from one buffer to another.

Checking Buffer Status

The *buffer status command* **<ctrl-X><ctrl-B>** can be used when you are already displaying more than one window on your screen.

When you want to remove the buffer status window, use either the *one window* command **<ctrl-X>1**, or move your cursor into the buffer status window using the *next window* command **<ctrl-X>N** and replace it with another buffer by typing the *switch buffer* command **<ctrl-X>B**.

Saving Text From Windows

The final step is to save the text from your windows and buffers. Close the lower two windows with the *one window* command **<ctrl-X>1**. Remember, when you close a window, the text that it displayed is still kept in a buffer that is *hidden* from your screen. For now, do *not* save any of these altered texts.

When you use the *save* command **<ctrl-X><ctrl-S>**, only the text in the window in which the cursor is positioned is written to its file. If only one window is displayed on the screen, the *save* command will save only its text.

If you made changes to the text in another buffer, such as moving portions of it to another buffer, MicroEMACS would ask

```
Quit [y/n]:
```

If you answer '**n**', MicroEMACS will *save* the contents of the buffer you are currently displaying by writing them to your disk, but it will ignore the contents of other buffers, and your cursor will be returned to its previous position in the text. If you answer '**y**', MicroEMACS again will save the contents of the current buffer and ignore the other buffers, but you will exit from MicroEMACS and return to the shell. Exit from MicroEMACS by typing the *quit* command **<ctrl-X><ctrl-C>**.

Keyboard Macros

A *keyboard macro* is a set of MicroEMACS commands that are stored in memory and given a name. After you create a keyboard macro, you can execute it again and again just by typing its name. If you are revising a big file, you will find that keyboard macros are one of the most useful features in MicroEMACS, and one that you will use often.

The following table summarizes MicroEMACS's keyboard-macro commands:

<ctrl-X>(	Open a keyboard macro
<ctrl-X>)	Close a keyboard macro
<ctrl-X>E	Execute a keyboard macro
<ctrl-X>M	Rename a keyboard macro
<ctrl-X>I	Bind current macro as initialization macro

Creating a Keyboard Macro

To begin to create a macro, type the *begin macro* command **<ctrl-X>(**. Be sure to type an open parenthesis '(', not a numeral '9'. MicroEMACS will reply with the message

```
[Start macro]
```

Type the following phrase:

```
MAXNUM
```

Then type the *end macro* command **<ctrl-X>)**. Be sure you type a close parenthesis ')', not a numeral '0'. MicroEMACS will reply with the message

```
[End macro]
```

Move your cursor down two lines and execute the macro by typing the *execute macro* command **<ctrl-X>E**. The phrase you typed into the macro has been inserted into your text.

If you give these commands in the wrong order, MicroEMACS warns you that you are making a mistake. For example, if you open a keyboard macro by typing **<ctrl-X>(**, and then attempt to open another keyboard macro by again typing **<ctrl-X>(**, MicroEMACS will say:

```
Not now
```

Should you accidentally open a keyboard macro, or enter the wrong commands into it, you can cancel the entire macro simply by typing **<ctrl-G>**.

Execute a Macro Repeatedly

As with most MicroEMACS commands, you can use a keyboard macro with an argument to execute it repeatedly. For example, if you have defined a keyboard macro, then typing

<ctrl-U><ctrl-X>E

executes that macro four times. (Remember, four is the default value for **<ctrl-U>**.)

As described above, **<ctrl-U>** normally is used with a positive number, to indicate how often MicroEMACS should execute a given command or macro. With keyboard macros, however, you can use a special value for **<ctrl-U>**: -1. This tells MicroEMACS to repeatedly execute a keyboard macro until it fails.

For example, consider that you define the following keyboard macro:

<ctrl-S> foo **<ctrl-K>**

This macro searches for the string “foo”, then kills the rest of line that that string is on. Now, when you type the command

<ctrl-U> -1 **<ctrl-X>E**

executes this macro until MicroEMACS can no longer find the string “foo”; it then quits.

Obviously, you should define your macro carefully before you execute it with this -1 option to **<ctrl-U>**; otherwise, you can commit tremendous mayhem on your file with one keystroke.

Replacing a Macro

To replace this macro with another, go through the same process. Type **<ctrl-X>(**. Then type the *buffer status* command **<ctrl-X><ctrl-B>**, and type **<ctrl-X>)**. Remove the buffer status window by typing the *one window* command **<ctrl-X>1**.

Now execute your keyboard macro by typing the **execute macro** command **<ctrl-X>E**. The *buffer status* command has executed once more.

Renaming a Macro

Many times during a long editing session, you will find that you use one keyboard macro, then use a second keyboard macro, then find that you need the first macro again. In previous releases of MicroEMACS, the only way to do this work was to type the first macro, replace it with the second macro, then retype the first macro when you need it again. The present edition of MicroEMACS, however, lets you define any number of keyboard macros, and save them by giving each one a unique “name” that is, its own unique keyboard binding.

To rename a keyboard macro that you have already created, use the *rename macro* command **<ctrl-X>M**. To see how this works, do the following: (1) Type **<ctl-X>(** to open the keyboard macro. (2) Type **<esc>S** xyz **<ctrl-U> <ctrl-D>** to fill the macro with something. (3) Finally, type **<ctrl-X>)** to close the macro.

Now, type **<ctrl-X>M**, to rename the macro. MicroEMACS will reply with the prompt:

```
enter keybinding for macro
```

Type **<esc>L**. This tells MicroEMACS to take the keyboard macro you created and link it to the keystrokes **<esc>L**.

Now, whenever you type **<esc>L**, MicroEMACS will execute **<esc>**s xyz **<ctrl-U> <ctrl-D>**. You can now define another keyboard macro without wiping out the one you have renamed. There is no theoretical limit to the number of keyboard macros you can create, although there are practical limits imposed by the amount of memory available to MicroEMACS.

Renaming Macros: A Few Caveats

Please note that if you name a keyboard macro with a keystroke that is already defined, MicroEMACS will no longer be able to access that keystroke's functionality.

For example, if instead of naming your new macro **<esc>L** you named it **<ctrl-A>**, then every time you typed **<ctrl-A>** MicroEMACS would execute **<esc>S** xyz **<ctrl-U> <ctrl-D>** and you would no longer be able to jump to the beginning of a line (which **<ctrl-A>** normally does).

The only exceptions are **<ctrl-X>**, **<esc>**, and the **<ctrl-X>R** command (described below), which MicroEMACS will not let you reassign. Obviously, you should be very careful when you assign a name to a keyboard macro, or you could easily clobber much of the editor's functionality.

Note, too, that MicroEMACS lets you define reflexive keybindings, but these never work. For example, if you named the above example macro **<ctrl-D>** instead of **<esc>L**, then every time you typed **<ctrl-D>** MicroEMACS would try to execute a macro that included **<ctrl-D>** in it. Obviously, this can tie MicroEMACS into knots in no time. Again, please be very careful when you assign names to keyboard macros.

The commands **<ctrl-X>S** and **<ctrl-X>L** let you save all named keyboard macros into a file, and restore them during a later editing session. These commands are described in the next section.

Setting the Initialization Macro

MicroEMACS allows one macro to be specified which will be executed every time MicroEMACS is invoked. This "initialization macro" can be set using the key sequence **<ctrl-X>I** and causes MicroEMACS to "bind" the currently defined macro to the initialization macro.

Flexible Key Bindings

As you have noticed by now, MicroEMACS works through standard *key bindings:* that is, one keystroke or combination of keystrokes tells MicroEMACS to perform a particular task. For example, typing **<ctrl-A>** tells MicroEMACS to move the cursor to the beginning of the line; typing **<ctrl-E>** tells MicroEMACS to move the cursor to the end of the line; and so on.

MicroEMACS allows you to change its key bindings, so you can bind a given keystroke or combination of keystrokes to a task other than the default one documented in this tutorial. In this way, you can reconfigure MicroEMACS so that it resembles another editor with which you are more familiar.

To perform this magic, MicroEMACS uses two tables for keybindings: a *default table* that is loaded at compile time and never changes, and a *dynamic table* that you can modify with MicroEMACS's keybinding commands.

The following table summarizes MicroEMACS's commands for flexible keybindings:

<ctrl-X>R	Replace one binding with another
<ctrl-X>X	Rebind metakeys
<ctrl-X>S	Save flexible bindings and macros into file
<ctrl-X>L	Load flexible bindings and macros from file

Changing a Keybinding

The *replace binding* command **<ctrl-X>R** replaces one binding with another. For example, if you wished to replace the *beginning of line* command **<ctrl-A>** with **<esc>Z**, you would do the following:

1. Type **<ctrl-X>R** to invoke the rebinding command.

2. When you see the prompt

```
Enter old keybinding
```

type the keybinding you wish to change in this case, **<ctrl-A>**.

3 When you then see the prompt

```
Enter new keybinding
```

type the keybinding to which you wish to change it in this case, **<esc>Z**.

Note that you cannot rebind the command **<ctrl-X>R** itself; otherwise, you would paint yourself into a corner. Also, note that if you rebind a command to itself (that is, if you type the same keybinding in response to both prompts), then that keybinding is bound to the old meaning of the keybindings, should there be any.

Rebinding Metakeys

MicroEMACS's keybindings depend on several pre-defined metakeys. A *metakey* is a keystroke that introduces a further set of commands. MicroEMACS's default keybindings use two metakeys: **<ctrl-X>** and **<esc>**. Other editors use other keystrokes as metakeys. If you wish to rebind a metakey, use the *rebind metakey* command **<ctrl-X>X**. This command prompts you to bind up to three metakeys, and the argument key **<ctrl-U>**.

For example, suppose that you wish to change the metakey **<ctrl-X>** to **<ctrl-Q>**. To begin, type the command **<ctrl-X>X**. You will see the prompt

```
Enter prefix character 1 or space
```

“Prefix character 1” is **<ctrl-X>** in the default bindings. Type **<ctrl-Q>**. You will then see the prompt:

```
Enter prefix character 2 or space
```

“Prefix character 2” is **<esc>** in the default bindings. Since you do not want to change it, type **<space>**. You will then see the prompt:

```
Enter prefix character 3 or space
```

There is no “prefix character 3” in the default bindings, but you can set a third one for your keybindings if you wish. Since (for the sake of this example) you do not wish to set one, type **<space>**. Finally, you will see the prompt:

```
Enter repeat code or space
```

The “repeat code” executes a command repeatedly; in this tutorial, it is often called the “argument key” or “argument command”. Since (in this example) you do not wish to change it, type **<space>**.

Now that you have reset the **<ctrl-X>** metakey, you must now type **<ctrl-Q>** every time in place of **<ctrl-X>** throughout all of the MicroEMACS commands. For example, if you wished to change the metakey back from **<ctrl-Q>** to **<ctrl-X>**, you would have to type **<ctrl-Q>X** to invoke the *rebind metakey* command.

Note that because **<ctrl-Q>** already is bound in the MicroEMACS keybindings, when you rebind it the command to which it was bound is no longer available to you. However, if you un-rebind the key, then it automatically is rebound to its old command. In the above example, **<ctrl-Q>** is bound to the *insert literal character* command, which lets you insert control characters into your file. When you rebound the **<ctrl-X>** metakey to **<ctrl-Q>**, then the *insert literal character* command was no longer available to you. However, when you re-rebound the **<ctrl-Q>** metakey to **<ctrl-X>**, then **<ctrl-Q>** was automatically rebound to the *insert literal character* command.

To change the first prefix character back to **<ctrl-X>**, type **<ctrl-Q>X**, then enter **<ctrl-X>** when you see the prompt for prefix character 1. This restores the original metacharacter. Note, however, that the original function of **<ctrl-Q>** (which is to let you embed control characters within a file) is no longer available to you: MicroEMACS “forgot” its original function when you made **<ctrl-Q>** into a prefix character.

Save and Restore Keybindings

MicroEMACS lets you save your rebound keybindings into a file, and reload them during another editing session. To save your keybindings into a file, type the *save keybindings* command **<ctrl-X>S**. Try it. You will see the prompt:

```
Store bindings file:
```

Type the name of a file. MicroEMACS then writes its keybindings into that file. This command saves all named keyboard macros that you have created. It also saves other aspects of the MicroEMACS environment that you have set; for example, if you have turn on word-wrapping, that fact will be saved.

To restore a set of keybindings, use the *restore keybindings* command **<ctrl-X>L**. Try it. You will see the prompt:

```
Load bindings file:
```

Type the name of the file in which you saved MicroEMACS' keybindings and all named keyboard macros. MicroEMACS will then load them into memory for you.

These commands let you prepare several sets of customized keybindings and macros. You can customize keybindings to suit your preference, or create custom sets of macros to suit any number of specialized editing tasks.

By default, MicroEMACS checks for the existence of file **$HOME/.emacs.rc** and executes it if found. You can generate a copy of **.emacs.rc** using the save-keybindings command **<ctrl-X>S**. The MicroEMACS command-line option **-f** lets you specify an alternate file of keybindings macros. After it load **.emacs.rc**, MicroEMACS then executes the initialization macro, if one exists. For example, if you wish to use the set of keybindings saved in file **keybind** to edit file **textfile**, then you would type the following:

```
me -f keybind textfile
```

As you can see, MicroEMACS's system of keyboard macros and flexible key bindings help make it an extremely flexible and powerful editor.

Sending Commands to COHERENT

The only remaining commands you need to learn are the *program interrupt* commands **<ctrl-X>! and <ctrl-C>**. These commands allow you to interrupt your editing, give a command directly to the shell, and then resume editing without affecting your text in any way.

The command **<ctrl-X>!** allows you to send *one* command line (one command, or several commands plus separators) to the operating system. To see how this command works, type **<ctrl-X>!**. The prompt **!** has appeared at the bottom of your screen. Type **lc**. Observe that the directory's table of contents scrolls across your screen, followed by the message **[end]**. To return to your editing, simply type a carriage return. The *interrupt* command **<ctrl-C>** suspends editing indefinitely, and allows you to send an unlimited number of commands to the operating system. To see how this works, type **<ctrl-C>**. After a moment, the COHERENT system's prompt will appear at the bottom of your screen. Type **time**. The COHERENT system replies by printing the time and date. To resume editing, then simply type **<ctrl-D>**.

If you wish, you can suspend MicroEMACS's operation, tell the COHERENT system to invoke another copy of the MicroEMACS program, edit a file, then return to your previous editing. To see how this is done, type **<ctrl-C>**. When the prompt appears at the bottom of your screen, type

```
me example1.c
```

It doesn't matter that you are already editing **example1.c**. MicroEMACS will simply copy the **example1.c** file into a new buffer and let you work as if the other MicroEMACS program you just interrupted never existed.

Exit from this second MicroEMACS program by typing the *quit* command **<ctrl-X><ctrl-C>**. Then type **<ctrl-D>**. Your original MicroEMACS program has now been resumed. However, none of the changes you made in the secondary MicroEMACS program will be seen here.

It is not a good idea to use multiple MicroEMACS programs to edit the same program: it is too easy to become confused as to which edits were made to which version.

The only time this is advisable is if you wish to test to see how a certain edit would affect your text: you can create a new MicroEMACS program, test the command, and then destroy the altered buffer and return to your original editing program without having to worry that you might make errors that are difficult to correct.

Now type **<ctrl-X><ctrl-C>** to exit.

Compiling and Debugging Through MicroEMACS

MicroEMACS can be used with the compilation command **cc** to give you a reliable system for debugging new programs.

Often, when you're writing a new program, you face the situation in which you try to compile, but the compiler produces error messages and aborts the compilation. You must then invoke your editor, change the program, close the editor, and try the compilation over again. This cycle of compilation editing recompilation can be quite bothersome.

To remove some of the drudgery from compiling, the **cc** command has the *automatic*, or MicroEMACS option, **-A**. When you compile with this option, the MicroEMACS screen editor will be invoked automatically if any errors occur. The error or errors generated during compilation will be displayed in one window, and your text in the other, with the cursor set at the number of the line that the compiler indicated had the error.

Try the following example. Use MicroEMACS to enter the following program, which you should call **error.c**:

```
main() {
        printf("Hello, world!\n")
}
```

The semicolon was left off of the **printf()** statement, which is an error. Now, save the file and exit from MicroEMACS; then try compiling **error.c** with the following **cc** command:

```
cc -A error.c
```

You should see no messages from the compiler because they are all being diverted into a buffer to be used by MicroEMACS. Then MicroEMACS will appear automatically. In one window you should see the message:

```
3: error.c : missing ';'
```

and in the other you should see your source code for **error.c**, with the cursor set on line 3.

If you had more than one error, typing **<ctrl-X>>** would move you to the next line with an error in it; typing **<ctrl-X><** would return you to the previous error. With some errors, such as those for missing braces or semicolons, the compiler cannot always tell exactly which line the error occurred on, but it will almost always point to a line that is near the source of the error.

Now, correct the error by typing a semicolon at the end of line 2. Close the file by typing **<ctrl-Z>**. **cc** will be invoked again automatically.

cc will continue to compile your program either until the program compiles without error, or until you exit from MicroEMACS by typing **<ctrl-U>** followed by **<ctrl-X><ctrl-C>**.

The MicroEMACS Help Facility

MicroEMACS has a built-in help function. With it, you can ask for information either for a word that you type in, or for a word over which the cursor is positioned. The MicroEMACS help file contains the bindings for all library functions and macros included with COHERENT.

For example, consider that you are preparing a C program and want more information about the function **fopen**. Type **<ctrl-X>?**. At the bottom of the screen will appear the prompt

```
Topic:
```

Type **fopen**. MicroEMACS will search its help file, find its entry for **fopen**, then open a window and print the following:

```
fopen - Open a stream for standard I/O
#include <stdio.h>
FILE *fopen (name, type) char *name, *type;
```

If you wish, you can kill the information in the help window and yank it into your program to ensure that you prepare the function call correctly.

Consider, however, that you are checking a program written earlier, and you wish to check the call to **fopen**. Simply move the cursor until it is positioned over one of the letters in **fopen**, then type **<esc>?**. MicroEMACS will open its help window, and show the same information it did above.

To erase the help window, type **<esc>1**.

Where To Go From Here

For a complete summary of MicroEMACS's commands, see the entry for **me** in the Lexicon. The COHERENT system includes three other editors: the stream editor **sed**, the popular screen editor **vi**, and the interactive line editor **ed**. Each can help you accomplish editing tasks that may not be well suited for MicroEMACS. For more information on these editors, see their tutorials or check their entries in the Lexicon.