Table of Contents

Disclaimer

The seller of the software described in this manual hereby disclaims any and all guarantees and warranties, both express and implied. No liability of any form shall be assumed by the seller, nor shall direct, consequential, or other damages be assumed by the seller. Any user of this software uses it at his or her own risk.

This product is sold on an "as is" basis; no warranty of merchantability is claimed or implied.

Due to the ill-defined nature of "fitness for purpose" or similar types of warrantees for this type of product, no fitness for any purpose whatsoever is claimed or implied.

The physical medium upon which the software is supplied is guaranteed for one year against any physical defect. If it should fail, return it to Mark of the Unicorn, and a new physical medium with a copy of the purchased software shall be sent.

The seller reserves the right to make changes, additions, and improvements to the software at any time; no guarantee is made that future versions of the software will be compatible with any other version.

The parts of this disclaimer are severable and fault found in any one part does not invalidate any other parts.

Installation Guide for Mince

Mince is a program that needs to know a variety of things about your terminal and system. There are also some things you have to do before you can start using Mince. The process of telling Mince about your system and performing these other tasks is called installation. This guide will help you go through the installation process smoothly and with as few problems as possible; we suggest you read the whole thing once before starting.

As always, but especially during the installation process, if you need any help, feel free to call or write. Our address is:

Mark of the Unicorn
P.O. Box 423
Arlington, Massachusetts 02174
617-489-1387

We will provide all of the help that we can in order that you successfully bring up Mince. If you are calling us, it would help tremendously if you track down the hardware manuals for your system and terminal and have them at hand; we may need to find out something from them in order to help you.

First Step: Backing Up Disks

The first step in installing Mince is to make a backup copy of the original distribution disk (the one we sent you), then put the original away (preferably in another room, if not another building) for safekeeping.

A backup copy is simply an exact, "image" copy of an "original" disk. There are two reasons for making backup copies. First, if you are using a disk heavily, a backup should be made periodically in case the main disk gets damaged physically or "bashed" (the information on it gets so scrambled that it is unusable). Second, if you happen to make a mistake when doing something, a backup copy allows you to recover from that mistake quickly. Don't be afraid to have lots of copies of your disks; disks aren't expensive but your data is, and there is safety in copies. (Remember, a computer is, among other things, a device for making more mistakes faster than ever before possible!)

Both reasons for making backups apply to Mince. If you make a mistake in the installation process, you can retrieve the original disk, make another copy, and start over. Second, after Mince is installed and in use, you will want to be able to recover in case a disk gets damaged or bashed.

Thus, to repeat: the first step is to make a copy of the original distribution disk. Then, use the copy for all of the following steps, and put the original disk away for safekeeping.

(The following discussion is only interesting if you are interested in a fine point or two.)

> The distribution disk is set up with a swap file and a Mince .COM file. By making copies of this disk, all other disks will have these as the first two files. It helps performance tremendously to make these the first two files on every disk you put them on.

(End of fine point.)

The Swap File

Mince uses something called a swap file. This file is a place to put parts of the documents that you are not currently editing (Mince "swaps" parts of your document to and from this file.) as well as a place for it to remember the specific details of your system. This file is called "MINCE.SWP" and there must always be a copy of it around that Mince can find.

The size of this file is important: it limits the total amount of text you can be editing at one time. A workable minimum is 24 pages or so, and the maximum size is 248 pages. Roughly, a 64 page swap file lets you edit files totalling up to 64K in size, but in fact the swap space is not completely utilized -- there are gaps -- and editing tends to increase the number of these gaps. For example, a 20K file will take up 20 pages when just read in, but might grow to 35 pages during a long and heavy editing session, even if you don't enter any new text. Thus, it is wise to leave yourself more space than you think you will need. We use a 64 page swap file and it seems to work quite well.

There is a swap file on the distribution disk that we sent to you. There will thus also be such a file on the copy of the distribution disk that you are currently using. You will not have to create another one unless you want to copy the Mince program onto other disks.

Second Step: The Configuration Program

Next, run the configuration program, "CONFIG". Config will lead you by the hand through the process of answering numerous questions. If you are in doubt about what to do, you should execute all of the steps from 1 to 8 in order.

Note that Config is a program that sometimes seems to be in a hurry. Many times you merely type the first letter of an answer (e.g., "y" or "n") and Config will fill in the rest of the word for you and go on from there. Other times, when you are told what the default answer is, you merely have to type a carriage return and that default answer will be used automatically.

The copy of the swap file that comes on the distribution disk (and is consequently on your backup) is set up for a Heath H19 terminal. If you have a different terminal, you must be sure to use step 2 of the configuration program so that you can tell Mince about it.

Third Step: What To Do If Something Goes Wrong

One possible problem is that the configuration program cannot write out the new version of the swap file. This will happen if you make the swap file larger than the space available on the disk. If the disk has other files on it, then you should find a blank disk and try again.

If the configuration program's terminal selection menu does not mention your terminal, you have an adventure ahead of you. You get to select the "not listed" option (option 1). Config proceeds to ask you lots of detailed questions about things that you might never have heard of. If you know the answers, fantastic! If not, try to look the answers up in your terminal manual, find a local expert, or call us. If you must call, it would help a great deal if you have the manufacturer's manual for your terminal at hand. If we had the manuals here, we would already have entered the terminal's characteristics!

If you use a video board as a terminal, you may or may not be in luck. If programs on your system can do cursor positioning just by "printing" special character sequences, then these can be entered to Config just like those of a standard terminal. However, if programs have to directly address locations in the screen memory, then you have a problem and will have to do some programming. If you have an Amethyst, it is annoying but straightforward to write a terminal abstraction which handles your board (see the Terminal Abstraction Documentation). If you don't have an Amethyst, you'll have to add character-driven cursor positioning to your BIOS.

Fourth Step: A Question of Taste

There are several questions that the configuration program asks you for which there are no correct answers; rather, they are about your personal preferences. The first such question concerns the size of the swap file, as mentioned above.

The next question is the preferred cursor line. This line is the line of the screen that the cursor (Point) will be displayed on after a View Next Screen, Refresh Display, or similar command is executed. We like it to be just above the center of the screen (not counting the two reserved lines at the bottom).

The tab spacing, fill column and indent column values can be easily changed while you are running Mince. Thus, you are selecting the values that you would like them to have when you first enter Mince, and you need not worry about one or two special cases where you might want them to be different.

First you will be asked to select the tab spacing. Mince has semi-variable tabbing: tab stops are set automatically a fixed number of columns apart (for example, if the spacing is eight, tabs will be set at columns 8, 16, 24, ...). ANSI (American National Standards Institute) does not have a standard for this, but recommends an eight column spacing; this is also the CP/M standard. We prefer a five column spacing, however, especially for program source code.

Next is the fill column. Several commands deal with left and right margins for the text. Examples are Fill Paragraph and Center Line. The fill column is the right margin and is the first column in which characters cannot appear. We have found a value of 65 to be reasonable, although both 70 and 75 have been used in various documents. (This text uses 65.)

Related to the fill column is the indent column. It is effectively the left margin and is the first column in which text can appear. A value of zero is normal and causes the text to be against the left hand edge. Other parts of the documentation use indented paragraphs and an indentation of five.

The last "taste" question concerns the delay count. There are some things Mince does when it sees that the user has not typed a character for a little while, like echoing prefixes (e.g., "Meta:") or writing pages through to the swap file (that "Swapping..." message). The delay count tells Mince how long a "little while" is. We find that about four seconds' delay before auto-swapping is reasonable and a value of 350 results in that much delay on our system. Doubling or halving the value doubles or halves the delay.

Fifth Step: Testing

Now that you have told the configuration program about your terminal, you should select option 6: terminal testing. This should clear the screen, print columns of '*'s on the left and right edges, from the bottom up, then '*'s along the top and bottom from right to left. The text 'This should remain' and

'This should go away' will appear on the screen near the top, and the latter should be erased almost immediately.

If the '*'s do not properly ring the screen either the cursor possitioning sequences are wrong or the size of the screen is improperly specified. If the text 'This should go away' is left on the screen the clear to end of line function is probably not working. In addition if some characters are missing or there are additional random characters, the padding might be wrong.

If the terminal test rolls the screen up one line, try reducing the number of columns by one (via option 5) and rerunning the test. If the test now works, you can restore the number of columns to what it was and Mince should work. The reason why the screen gets scrolled is that Config writes in the lower right character postion but Mince does not.

Remember to install your terminal definitions (select option 7) on the swap file so that Mince will be able to find them later.

Sixth Step: The Finishing Touches

Now that you have told the configuration program about your system, you should go into Mince and try it out briefly. Does Mince seem to work properly? Are the delays about right?

After you are basically satisfied that everything went smoothly (don't worry about the fine points too much), you should make a backup copy (on a third disk -- not the original!) of all of the work that you just did. Do not forget to delete any stray files that you created when testing Mince before making the backup.

(Another fine point.)

> If you have double density disks and would like to use them as double density, make the backup from the single density "master" to a double density "work" disk.

(End of fine point.)

You should now have three disks. The first one is the distribution disk. The other two are identical and contain the swap file as configured and Mince. Take the distribution disk and one of the copies and put them somewhere safe (and remove the "write enable" tabs).

You are left with one disk with a working Mince (of which you may want to make still more copies). We hope that you will enjoy using it.

This document is a series of lessons to teach the use of "Mince", a display-screen oriented text editor. The reader is assumed to have used a typewriter for the preparation of manuals, memos, or documents before, but never to have used a text editor. Additionally, the reader is assumed to already be familiar with logging on to the computer and typing at a computer terminal. Each lesson teaches a new set of commands or concepts about text editing in general and Mince in particular. There are eight lessons, on the following topics:

Lesson 1: Getting Started
This lesson teaches about the cursor, typing in text, and the commands used to move around and delete and insert text.

Lesson 2: Moving Around Faster
This lesson teaches commands for moving by words instead of characters, the universal repeat command, and beginning/end-of-line/sentence/buffer commands.

Lesson 3: Reading and Writing Files
This lesson explains the computer file system and how Mince uses it to store text. Commands for reading and writing files are demonstrated. At this point, the reader will be able to use Mince effectively for simple document preparation, although document modification may be somewhat time-consuming still.

Lesson 4: Searching
This lesson introduces the forward and reverse search commands.

Lesson 5: Keyboard Culture
This lesson explains a few minor but useful Mince commands. Some common abbreviations used in the other Mince documentation are explained to allow use of the reference manual.

Lesson 6: Killing and Moving Text
This lesson introduces the kill and yank commands. Extensive examples of text movement are given, since this is one of the more confusing aspects of Mince.

Lesson 7: Text Processing Commands
This lesson explains the command used for "filling" paragraphs of text, and the commands used to set parameters which control this operation. Commands which change the case of words are explained.

Lesson 8: Buffers
This lesson explains the effective use of multiple editing buffers.

Lesson 1: Getting Started

Mince is a text-preparation program for use with small computers. In many ways, it is similar to a very fancy office typewriter. However, it was designed to be used with a TV-screen computer terminal. During this first lesson, you should be getting used to typing on a computer terminal, if you haven't done some work on one already. The keys on each terminal or typewriter are placed a little differently; in particular, we will be making use of some of the out-of-the-way ones you don't usually hit.

So, let's get started. Sit down at the terminal attached to the computer, and type Mince's name to the computer. Type:

```
mince
```

and follow it with a carriage-return. (This is the key labelled "RETURN", "CR", or "ENTER" on your terminal.)

Mince will begin to run, and you will notice all sorts of things happening at once. There will be some buzzing over at the disc drives. This is where the computer stores Mince when it's not in use, and where you will store your text when you don't need it. (More about that in Lesson 3.) Your terminal screen will be blanked in preparation for your typing. A single line will appear at the bottom of your screen, something like:

```
Mince Version 2.5 (Normal) main: DELETE.ME  -0%-
```

which tells you that Mince is ready and waiting to go to work.

Let's start by just typing something on the keyboard. Type the four words:

```
This is a test.
```

Notice that as you typed, what you were typing appeared at the top of the screen. At the end of the line you typed, right after the period, is a solid or blinking box or underline. This object, called the "cursor", is not something you typed; it is just an indicator of where you are in your text. It functions just like the carriage of an ordinary typewriter, showing you where the next thing you type will appear.

The next thing you type will be a carriage-return. As on an electric typewriter, hitting the carriage-return key puts you at the beginning of the next line. After the carriage-return, type another line:

I am testing this text editor.

and follow it with another carriage-return. Now you have a two-line document.

Notice that the line at the bottom of the screen has not changed position throughout this typing. The entire screen of a terminal does not scroll the same way that a paper would in a typewriter. Only the top portion of the screen (20 lines or so) will scroll upward, if you type enough text. This area is called the "window". Imagine it as a small viewing area onto a large document, and the name will make sense. Let's type a little more, just to get used to it. Type:

It's not much different from using a typewriter.

and type a carriage-return.

We could go on typing lines of text, but sooner or later we would make a mistake. Let's make one now, on purpose. Type just the two words:

But typewriterb

and stop there. So much for our "error". The "b" in "typewriterb" should have been an "s". What do we normally do on a typewriter if we type a wrong letter? We use the erase key if it's a fancy office typewriter, else (yecch!) white-out. Mince has an erase key, too. Type the key labelled "DEL" or "DELETE" or "RUBOUT". (On some computer terminals you may have to hold down the shift key to get a delete. Make certain that you really are going to hit DELETE and not some other character.) Observe that the "b" in "typewriterb" has disappeared from the screen, and the cursor has moved backward a space, to right after the "r". Type an "s" now, and the word "typewriters" will become correct. It truly IS very much like using a typewriter.

Suppose we realize that we had forgotten to type some word before the word "typewriters". Just hit the DELETE key enough times to erase the word "typewriters". (Go ahead and do that now.) Now type:

MOST typewriters

and stop. The line now reads "But MOST typewriters" and the cursor is right after the "s" is "typewriters".

We had seen that all text we type is entered at the cursor;

now we've seen that text we delete is deleted at the cursor as well. This rule always holds true in Mince, and there's another demonstration of it coming up.

What if, as before, we discovered a word missing, only on the FIRST line this time? For example, let's suppose we wanted to change the sentence reading "This is a test." to "This is only a test." by adding the word "only". You certainly could type the DELETE key enough times to erase all the way back, but it would be an awful waste of time if we had to do it that way.

The way we modify or add text in Mince is to "go" to the place in the text where we want text to be added or changed, and then type in whatever text we want there. In order to "go" somewhere, we position the cursor to that place on the screen. To get to the first line of text, we will move the cursor up three lines, on at a time.

Look on your keyboard for a key labelled "CTL", "CTRL", "CONTROL", "CNTRL", or the like. This is a very important key for Mince; it is called the control key. It is like a shift key: you hold it down, and while holding it down, you type one or more normal characters. Like a shift key, pressing it and releasing it has no effect. When you press the "p" key (DON'T do it now), you get a lower-case "p". If you press the "p" key while holding the "SHIFT" key, you get an upper-case "P". If you press the "p" key while holding the "CONTROL" key, you get something called a "Control P". It is a different variety of P, just like p and P, except instead of being an upper case letter or a lower case letter, it is a called a "control character". All control characters are commands to Mince. (This makes the Control key very much like the "Code" key which is present on the fancy IBM typewriters.) You have seen what happens when you type normal upper and lower case characters; they go into the document. Control characters are used to control Mince, to manipulate the cursor and manipulate the text around it. (Note that although we will spell the control commands with a capital letter, it is not necessary to use the shift key AND the Control key to get the command. "Control P" is just the same as "Control p".)

To make Mince move the cursor to the previous line, we use the control character "Control P", the P being for "Previous". Do so: hold down the "CONTROL" key, like a shift key, and press the "P" key, while still holding the "CONTROL" key, then quickly release both. Now look at the screen: you will observe that the cursor has indeed moved up to the previous line, to a place right above where it had been.

Now let us go up two more lines, to the first line. Hold down the control key again, and type "P" twice while holding it. Again, the cursor will jump up a line each time.

Now that we are on the correct line for the change we wish to make, we can move around on that line to get to the correct

place. We can move backwards on that line by telling Mince to go backwards: this is done with a "Control B", B for "Backwards". Hold down the control key. Now watch the screen and press the "B" key several times, while still holding the control key. You will see the cursor move backwards, that is, to the left, one character position for each time you press the "B" key. Soon you will reach the beginning of the line, at which time you cannot go any further back, because that is the beginning of your text.

We now want to move forward to the place after the word "is", in order to insert the word "only". Hold down the control key, and type the letter "F" several times. "Control F" is the Mince command for "forward", i.e., move the cursor forward one character. While still holding down the control key, type F's until the cursor is under (or covering, on some terminals), the word "a". If you type the Control F too many times, simply type Control B's until you get to the right place. What you are doing now is the most important form of interaction with Mince - issuing commands until you are "at the right place", or "the right thing has happened", doing one command at a time and observing its effect, and repeating that process until you have achieved what you want.

Now the cursor is at the "a", and we wish to put the word "only" there. That is trivial: simply type the four letters o, n, l, and y. You will watch Mince move the rest of the line over to make room for the new text and you now have on that line:

This is onlya test.

with the cursor still under the "a". Immediately, we perceive a problem: there is no space between "only" and "a". This is simply because we did not enter one. Hit the space bar. Now we have:

This is only a test.

with the cursor still on the "a". We have now fixed the text we wished to. Note that in order to type in the new word, "only", we did not have to say anything special, we just moved the cursor to the right place, and started typing. Whenever we type a non-control character, it goes into the text at the cursor, and moves the cursor over one to the right. If you think for a little while, you will notice that that is just what it did when we were simply typing in the lines on the screen! It does look a little unusual (and not at all like a typewriter) to have an entire line moving to the right, but it's easy to get used to.

Now we must get back to the end of the document, where we had left off when we decided to add the word "only" to the first line. We can do this by going to the Next line, and the Next line, until we are where we want to be. We do this with the "Control N" command ("N" for "Next", as you might have guessed). Hold down the control key (As you have seen, if you do not the

characters you type will go into your text!), and press "N" three times. Each time, the cursor will move one line down. We are now on the right line. You will find that you are in the middle of the line, because Mince tries to keep you in the same "column" when going between lines. Type a few Control F's to get to the end of the line. Again, you will notice that at the edge of your text, the cursor stops, because you can go no further forward.

Now let's finish the sentence "But MOST typewriters are old-fashioned." Enter the words:

are old-fashioned.

and surely enough, your entire text appears correct before you:

This is only a test.
I am testing this new editor.
It's not much different from using a typewriter.
But MOST typewriters are old-fashioned.

If you haven't already done so, type a carriage-return after the last sentence. Now the text has five lines in it, and the last one is a blank line. Now type a "Control P" to move up to the previous line. The cursor is now at the "B" in "But".

Now type a "Control B". Notice that the cursor has moved to the end of the previous line, right after the period! This is not what we would expect from a typewriter, where the carriage would just have bounced off the left margin. In Mince, however, the carriage-return you type is turned into a "newline" character, which is inserted in the text. On the terminal's screen, this newline character looks like just that: a new line. But newline is a character, too, just like all the others you have been typing in the text. Since carriage-return or "newline" is a character just like any other, it can be moved across, just like any other character. Type a "Control F" now. As you probably expected, the cursor moves back to the beginning of the next line. Now type the DELETE key. We can delete the newline characters as well, just like any other characters. Now the text window should have:

This is a test.
I am testing this text editor.
It's not much different from using a typewriter.But MOST typewriter
old-fashioned.

Notice that the third line of text has gone past the edge of the terminal's screen, and the rest of the line appears on the next terminal line. (Probably the word "old-fashioned" is split across the two screen lines.) Just because there are two screen lines does not mean that there are two real lines of text there, separated by a real carriage-newline character, though. You can usually tell the difference by the fact that one line runs off to

the very last column on the screen. You will not usually type such long lines, just as you would not type off the end of the piece of paper in a typewriter carriage. It is possible, however, to get lines to join together and create a longer line, as we just did. Insert the newline back again: type carriage-return and it gets inserted in front of the cursor, just as we would expect. Insert another newline, so that there is a blank line inbetween the third and fourth (now fifth) lines of text.

What if we wanted to change "But MOST typewriters" to "Wood stoves"? We have to erase the first three words, then insert the new text. So far, the only way you know to delete text is to use the DELETE key. But if you were to hit it now, it would delete the character you just entered, that is, the newline. We could, of course, type some Control F's to get the cursor past the words, and then type some DELETEs. (Or we could remove the words one character at a time, by alternately typing Control F and DELETE.) But this would be cumbersome. Instead, Mince has a command to delete the next character in the text rather than the previous one: "Control D". Hold down the control key and type "D" enough to get rid of the words "But MOST typewriters". Notice that, as usual, the entire line is "eaten up" and moves to the left as the characters disappear.

Now that the words are gone, type the DELETE key, and the blank line we inserted earlier will also disappear. We see that the DELETE key erases characters to the left of the cursor and the Control D command erases characters in the other direction, that is, it erases the character that the cursor is on. Now type "Wood stoves" to insert those words at the beginning of the line. Type a space after "stoves" if you happened to delete the one to the right of "typewriters" earlier.

Take a quick look at the "mode line" at the bottom of the screen again. The percentage mark is no longer zero, as it was when we started. This number just says roughly how far through the file the cursor is. It's not too useful when you can see your entire document on the screen as we can now, but for large gobs of text, it's handy to know where you are. Also, there is a star at the right edge of the mode line which appeared there when you started typing. This asterisk means that the text which is on the screen is different from what you started with (in this case, nothing at all). It turns on in order to tell you that your text has not been saved anywhere, and that if you leave Mince without saving it, your work will be lost. We'll learn about saving the text we type in Lesson 3.

You may want to continue experimenting with the things you have just learned: when you are done, you will have to leave the editor: this is done as follows: type a "Control X" followed by a "Control C" (C for "Command level"). This is not an easy command to type or remember. That is reasonable, though, because you may be editing for hours, and you will only type it once. Hold down the control key, and type an "X" and a "C" while holding it.

Mince will respond:

Abandon modified buffer(s)?

because you have done work that you have not saved. That's O.K. for now. Type "Y". Mince will leave you at command level, at the bottom of the screen.

We just learned the six most important Mince commands:

DELETE	Erase the last character.
Control D	Erase the next character.
Control F	Go Forward one character.
Control B	Go Backward one character.
Control N	Go to the Next line.
Control P	Go to the Previous line.
Control X Control C	Quit the editor.

We have also learned some basic Mince vocabulary:

cursor:	The solid or blinking box or underline which shows where you are about to insert or delete text on the screen.
window:	The area which covers most of the screen, and which displays twenty to twenty-five lines of the text which you are currently editing. This window moves around so that the cursor is always somewhere in the window, regardless of how you move it while inserting or deleting text.
mode line:	The line at the bottom of the screen which tells you that (1) you are talking to Mince, (2) how far through the text you are, and (3) whether the text on the screen has been changed since it was last saved. There are quite a few other pieces of information which the mode line gives; more about those in Lessons 3 and 8.

Practice using the commands in this lesson until you are thoroughly familiar with them. They are the most important and useful commands. With these commands, and one or two others, you can do just about anything you will ever be called upon to do. All the other commands just make it easier to do more complex things, but you can always move around and type in text with these.

Lesson 2: Moving Around Faster

To start this lesson, enter Mince as you learned in Lesson 1. Type the following text in to Mince. Since it is poetry of a sort, be sure to break the lines (type carriage-returns) at the proper places. Use the DELETE key to correct some of the mistakes as you type it in, but don't be too careful about the words' spelling -- the poem is intentionally nonsense. The text:

'Twas brillig, and the slithy toves
Did gyre and gimble in the wabe.
All mimsy were the borogoves,
And the mome raths outgrabe.

"Beware the Jabberwock, my son!
The jaws that bite, the claws that catch!
Beware the Jubjub bird and shun
The frumious Bandersnatch."

by Lewis Carroll

Now let's go back and make certain that the text IS letter-perfect. Do you have blank lines between the verses and before and after the author's name? This should be a good chance to review using Control P (Previous line), Control N (Next line), Control B (Backward character), Control F (Forward character), and Control D (Delete character). It won't take long to make it perfect -- it's easy to turn a draft into a final if you don't have to retype the whole thing!

As an aside... How are you typing the control characters? Using the index finger of one hand to hold down the control key and using the index finger of the other hand to hit the letters? That's the way most new users (and programmers!) do it. But remember that you can go much faster if you learn to touch-type the control keys, too.

Well, as usual, we are going to change the text you just typed in and perfected. (With any luck, after we're done, the first verse will make a little more sense!) Before we start, make certain that you typed a newline as the last character, and move to the end of the text. The cursor should be directly under the "b" in "by Lewis Carroll". Changing this verse could be a

real chore if all we had were the commands we learned in Lesson 1. Moving one character at a time is slow and fairly annoying, especially if we can see the place we want to get to to make a change.

For example, how would you change the last word on the previous line? Typing Control P and lots of Control F's could take a while. (If you're a smarty, you have realized that it would be easier to just type a Control B and pass over the newline character! But then what about the last word on the second line above that?) To illustrate a new command, let's change the word "Bandersnatch" to "Sandpiper". Type some Control P's to position yourself on the proper line. Now type a Control E. Control E stands for "go to the End of the line", which you should have seen the cursor do! Now type in enough DELETEs to erase the word "Bandersnatch" (Perhaps you will want to type some Control B's to pass over the punctuation first.) and then type in "Sandpiper".

Now, suppose we wanted to change the word "The" at the beginning of this line to the word "A". We could type some Control B's, but this would take a few too many keystrokes. Instead, there is a Mince command which moves to the beginning of the line: Control A. It doesn't stand for anything in particular, but an easy way to remember that a Control A moves to the beginning of the line is that "A" is at the beginning of the alphabet, or that it is on the far left of the keyboard and the cursor moves to the far left of the line. So, go ahead now and type a Control A, then type three Control D's to delete the word "The" and then insert the word "A". Now type enough Control N's to get back to the bottom of the text. Notice that if you keep typing control N's, you still stop at the last line you entered. Similarly, if you type Control P's when you are at the beginning of your text, you will just stay in the same place. (Similarly for Control B's at the beginning and Control F's at the end.)

Now that we're at the end of the text, what if we noticed that we wanted to change the line that says "All mimsy were the borogoves"? We could type a lot of Control P's, one at a time. This is easy enough but may be more time-consuming for longer "distances". The command Control U (for "Universal repeat count") will allow us to make the jump with less typing. Type the Control U. You will notice a message at the bottom of the screen saying "Arg:" appear, along with the number "4". Type a "9", (The number in the message will change to "9" also.) since the line we want is nine lines above where the cursor is now. Then type a Control P. Nine Control P commands in a row will automatically get executed. If we decided that this wasn't where we wanted to be, we could have typed "Control U 9 Control N", and gone down nine lines just as easily. Control U repeats anything. Anything? Yes. Type "Control U 10 *" (that is, type the Control U, then type the numeral one, then the numeral zero, then an asterisk). Ten stars will appear in your text. Now type a Control U, then type the number ten, and then type the DELETE

key. They will all go away again. Control U really does repeat anything.

Why did the number four appear as the "Universal" repeat count each time before we typed in our number? If we hadn't typed a number, whatever character (control or otherwise) typed subsequently would have been repeated four times. This is just for convenience: for example, it is easier for most people to type "Control U, Control P, Control P, Control P" to move back six lines in the text than it is to reach out to type the number six after the first Control U. This isn't quite so true for large numbers, so Control U has another unusual property: it multiplies any other previous universal repeat given! This means that if we type Control U in front of another Control U, the number of times whatever command we type next will be repeated is 16. In the same manner, "Control U 7 Control U" will repeat the next command 28 times. This can be quite handy, since "Control U Control U Control U Control N" moves the cursor about two thirds of a screenful on most terminals. (That sequence is typed about as often as "Control U Control N Control U Control N", which moves down eight lines. You really can almost always avoid typing a number for the Universal repeat!) Try that now, on the line we are on: type "Control U Control U Control F", to move the cursor forward sixteen characters.

We can jump across characters faster than before just by using the universal repeat command. But it would be much more convenient to be able to move across words without counting how many characters there are in each of them. For that, we will have to learn about a new key on the keyboard and a way to give Mince commands other than the control key.

Search around on your keyboard for a key labelled "ESC", "ESCAPE", "ALT", or "ALTMODE", or something like that. This is called the "ESCAPE" key, which generates a character called the "ESCAPE character". We will refer to it by the letters ESC; this means strike and release the ESCAPE key, do not type the letters E, S, and C! This key does not work like the Control key; you do not hold it down while typing characters. Just type it and then type another character after it. For example, ESC F would be typed by first hitting the Escape key, then hitting the letter "f".

Try that now. Type an "ESC F" and see what happens on the screen. Try it again a few times. The ESC F command stands for "move Forward a word". Let's change the word "brillig" to "overcast". Move to the top line of the screen (in whatever method is most convenient for you now), to the first line of text, on the left-hand edge (use Control A to do this). Use the ESC F command to skip past the word "'Twas", and then use it again to skip past the word "brillig". Two interesting things happened when you typed those ESC F's. On the first one, we had punctuation in front of the word; the ESC F command doesn't care, and just skips to the end of the word anyway. On the second one,

the ESC F stopped before going past the comma; this is because the punctuation is not part of the word. What the ESC F command really does is to move to the end of whatever word is in front of it. If we were in the middle of a word and typed ESC F, it would go to the end of that word. If we were five lines above a word, but all five lines were blank, ESC F would still put us at the end of that word.

To delete the word, we could type many DELETE's. But as you might have guessed, there is an ESC command to do that, too. To delete words, use the ESC DELETE command. Type the Escape key, then type the DELETE key, and the word "brillig" will disappear. Now type in "overcast".

Let's also change the word "toves" to "foxes". To do that, move past the next three words (only THREE, the words "and", "the", and "slithy") with the ESC F command. Now let's delete the word "toves", by using the ESC D command. As you can now see, there are parallel commands for Control F (ESC F), DELETE (ESC DELETE), and Control D (ESC D), all of which work on words rather than characters. Now that we've deleted the word with ESC D, let's type in the word "foxes".

But look at the screen! Since we didn't type a space before the next word (or maybe you noticed that there wasn't one, and did!), the last two are run together. Why did the space between the words get deleted with ESC D, but not with ESC DELETE before? Just like the ESC F command, the ESC D command only knows about deleting to the end of the next word. So, if you are in front of a space which itself is in front of the next word, ESC D will delete the space, too, since all Mince does is to delete until it reaches the end of the next word. So, in the example mentioned above with five blank lines and then a word, if we had typed ESC D rather than ESC F, we would have deleted the five blank lines, and the word following them.

Let's fix up the missing space by typing Control B's until the cursor is on top of the "f" in "foxes", and then typing a space. You may have noticed that we didn't mention an ESC B (to go with Control B) above. Well, surely enough there is one. Type that now, and notice that the cursor jumps backward across the word "slithy". Let's delete that word (either with Control D's or ESC D) and type in the words "quick brown". Just so you know that you've gotten everything right this lesson, the first line should now read "'Twas overcast, and the quick brown foxes". If it's not correct, take a minute to fix it up now using some of the commands you've learned in this lesson.

Now let's change the word "wabe" to "swamp". One easy way to do this would be to notice that the word "wabe" is at the end of the next line, almost below where we are now. It could be reached by a single Control N. Another way would be to see that it is the eighth word after the ones we just typed in, and that it can be reached by typing Control U 7 ESC F. However you get

there, delete the word using either ESC D or ESC DELETE, and insert the word "swamp". If you used the ESC DELETE command, you might have to type the period at the end of the line in again, since it may have been eaten up in the backward deletion of the word.

Suppose we wanted to insert the name of the poem, "Jabberwocky", in the line with the author's name? We could type some Control N's to get to the last line, but there is a quicker way. Type "ESC >" (the ESCAPE key, then a greater-than symbol). This, as you see, puts the cursor at the end of all the text. Now type a Control P, to position the cursor at the beginning of the proper line, and insert the title and a comma. There is a command to move as far as possible in the other direction, too. Type "ESC <", and you will move the cursor to the beginning of the text.

There are two more quick-motion commands which you may or may not want to practice using. These are ESC A and ESC E. They are similar to Control A and Control E, except that they move to the beginning and end of sentences rather than lines. Now we have a reasonably complete set of parallels for the control commands we learned in Lesson 1:

Control F	Move forward a character
ESC F	Move forward a word
Control B	Move backward a character
ESC B	Move backward a word
Control D	Delete forward a character
ESC D	Delete forward a word
DELETE	Delete backward a character
ESC DELETE	Delete backward a word
Control A	Move to beginning of line
ESC A	Move back to beginning of sentence
Control E	Move to end of line
ESC E	Move forward to end of sentence
ESC <	Go to beginning of text
ESC >	Go to end of text

Play around with these commands a little while, until you are used to what they do and are able to remember most of them off the top of your head, without looking at this page. We haven't yet succeeded in making sense out of the first verse; now's your opportunity!

When you are done and want to leave the editor, type

Lesson 3: Reading and Writing Files

In this lesson, we won't be using Mince at all. (How refreshing, eh?) We'll just be chatting about computer "files". At the end of this lesson, you can try out the commands you learn, and you will (finally) know all the commands you need to do some "real work" using the editor.

On to files... A computer's filing system is pretty much the same as the one you or I use. If you want to look at an old report, what do you do? You pull out the file from the filing drawer and put it on your desk. To find a particular spot in the report, you page through it on your desk. You might change it or add something to it, then put it back into the file drawer.

The operations are similar on the computer system. The only differences are the ways in which the "file drawers", "files", and "pages" work, and in the names computer scientists decided to give them.

Let's think of the disc drive as the filing cabinet and each of those flexible "floppy discs" as a sort of removable filing drawer. The unfortunate property of these discs is that they are only useful when put into the disc drive, which is not true of filing drawers and cabinets! On each of the discs, as in each of the drawers, are many files. You can see what files you have in a drawer either by leafing through them, or by looking at the short label on the front of the file drawer. There is a label space on each flexible disc which is similar to the one on the whole drawer, but how can you find out the names on each of the files? There is a computer program to do this for you. Just try typing "DIR" to the computer system when you aren't using Mince, and it will list the names of the files. Most files have two names, like "PRGINTRO.DOC", that is, names with some characters, then a dot, and then some more characters. The first bunch of characters can't be longer than eight, and the second bunch can be up to three characters long. This is a lot more limited than what you can write on a file folder, so you can now understand why some computer files seem to have such cryptic names.

What is the analogy of getting the file out and putting it on your desk? To Mince, this is called "reading" a file. Mince has an unusual way of manipulating it, though. Just as you would probably not mark up the originals of a report in your file

folder, Mince does not change the contents of the file on the floppy disc. You would make a photocopy of your work, mark it up or change it, and then perhaps retype it and replace what was originally in the file folder. Mince does something similar: it "reads in" a copy of that file, and then lets you modify that copy. This has one unique drawback and one unique benefit: if you make a mistake, you can just quit, and you haven't actually wreaked havoc on the original file; on the other hand, if you don't remember that you must save what you've done back in the file, you can lose all your hard work.

Therefore, there is a command in Mince to save the work that you've been doing as the original again. This is analogous to putting your new work back into the file, and back into the file drawer. To Mince (and computer scientists), this is known as "writing" the file. (It surely isn't what you or I think of when someone says "write the file out". "Really," you would say. "Longhand?")

In Mince, the part which performs the function of the working desk while we manipulate the file is the "buffer". Files are "read in" to the buffer, and they are "written out" back to the computer filing system. The computer terminal screen is a sort of window into that buffer, and we can move the window around with the control commands you have been learning recently. Just as paper files have pages, our computer file has pages, which we can see by issuing "page" commands to Mince.

Enough generalizing. Here are the commands:

Control X Control R:	Read the file from the computer filing system into the Mince buffer.
Control X Control W:	Write the file out from the Mince buffer back into the computer filing system.
Control V:	View next page of the file in the buffer.
ESC V:	View previous page of the file.

Actually, the Control V and ESC V commands are misnomers. They really have nothing at all to do with pages of text as we would ordinarily see them on a desk. They just serve to move the window so that most of it rests on what was unseen text, either before or after the windowful that is currently in the window. After all, most paper pages have 55 to 60 lines of text, and our terminal screen window has only 20 or so lines. In order to provide some continuity between screenfuls, these commands will overlap pages: one or two lines at either end of the screen will be from the next screenful of text in that direction. The View Screen commands take Universal repeats, just like other commands, so that "Control U 5 Control V" will move the window "down" in

the text by five screenfuls, or about 100 lines. These commands do not do anything that can't be done with Control P and Control N; they just do it a little faster.

The Control X commands are a bunch of commands for which there was no room in control keys for, or commands which we want to be absolutely certain we want to type, so that there can be no mistake about what we are doing. Remember the "Control X Control C" command? Now that you know it has the possibility of throwing you out of Mince without your having saved the work you've been doing, aren't you glad it's just a little harder to type? (Remember back in Lesson 1 when we typed Control X Control C? Mince asked us if we wanted to "abandon" a "modified buffer". Now you know what that's all about!) The easiest way to remember what the Control X means is to imagine that these specialized commands are part of the "eXtended" command set.

The "Control X Control R" command, in order to read a file, must know which file you want to read. When you use it, it will ask you, in the echo line (the last line on the screen), for the name of the file you wish to read into Mince's buffer. Just type in the file name, followed by the carriage-return key. Don't worry about remembering all this -- when you type "Control X Control R", Mince will print out a message asking you to type in the file name and reminding you that you have to enter it with a carriage-return. It looks something like:

File to Read <CR>:

Similar things happen with the "Control X Control W" command. In general, we will want to write out the buffer back to the same file we read it in from, but occasionally, we will want to save it in a new file. (This is particularly true of a form letter that we modify slightly for a particular recipient, and do not want to change the original for.) After you type "Control X Control W", Mince will say:

File to Write <CR>:

You can again type in the name of the file into which the text is to be written, followed by a carriage-return. If you make mistakes in the name of the file, you can use the DELETE key, just as you would in the editor.

Since writing files back out to the same place they were read from is such a common operation, Mince allows you to avoid typing the entire file name each time you give the Control X Control W command. Rather than typing out the file name, just type a carriage-return. Mince will write the buffer out to the computer filing system under whatever name was last used in either a Control X Control R or Control X Control W command. How do you know where it will go without always remembering the file name you last used? Mince displays that name in the "mode line", which we learned about in Lesson 1. It looks something like:

Mince Version 2.5 (Normal) main B:PRGINTRO.DOC -0%-

and the phrase "B:PRGINTRO.DOC" says what the default file read/write name will be if we do not specify it; that is, if we just type a carriage-return in response to any of the file name questions. (You may have noticed the "B:" alongside the file name. This is due to the fact that computers, like people, have many filing cabinets. Just as you would tell someone that a file is in the "right-hand" filing cabinet, you must tell the computer which disc drive to look for the file in or to store it in. The computer disc drives you will be using are named "A", "B", etc.)

Some kinds of files get pretty large. Computer files are no exception. Computer scientists just measure them in thousands of characters rather than in inches of paper. If you had a very large file to look at, you would never fit each page of paper side-by-side on your desk so that you could see all of them at any instant. Instead, you would probably keep 5 or so pages spread out on your desk, and keep the rest in stacks, occasionally removing or adding a page on a stack.

Similarly, many many thousands of characters cannot all fit into your computer at once. (Usually, Mince has room for ten to twenty thousand.) Therefore, your buffers of text may not be completely residing in memory at once. Mince has a special file on disc (not related to the other files we've been talking about) which is similar to the stacks of paper you would keep on your desk. As you use each new page (thousand characters) of the text buffer you are editing, Mince will pull it out of the "stack of pages" by reading it from its special "page file".

If you go back to old pages or add new text to pages, they will generally be in memory rather than on the stack in the page file. But, occasionally, the computer's memory will fill up, as would your desk if you kept looking at more and more pages. So, occasionally, Mince must write some of its old pages back to disc in order to make room for new ones.

You will know what is happening by the click or buzz at the disc drives. There will also be a message printed at the lower right of the screen" "Swapping..." which means that Mince is exchanging a page on disc for a page in memory. This message will go away when the swap is complete.

Why do you need to know all this? Well, occasionally you may have to wait for this page swapping to happen. Why? The computer stops listening to the terminal keyboard for a short time when it swaps pages. In particular, when you are just beginning to work on a buffer of text which you read in via a Control X Control R command, pieces of the buffer will be paged in as you need them. This means that Control V's will occasionally take a little longer than usual and produce the "Swapping..." message. Once read in, though, the pages you use

most (e.g. modify or go back to) will remain, and "page waits" will be less frequent. (Amazingly enough, computer scientists call this concept "paging" too! But a page of 1000 characters has little to do with a 55-to-60 line page of paper or a 20-line text window on a buffer!)

If you stop using the keyboard for a little while, Mince may spend a little time "cleaning up" while you are idle. It will swap out pages you have modified, so that any page swapping which is done while you are actually doing work later will occur a little faster. Don't worry about this too much; when you start typing on the keyboard again, the intermittent housecleaning noises will go away.

Don't worry about trying out the commands you have learned in this lesson, unless you have to start to work using the text-processing system right away. In the next lesson, you will get a chance to try out "Control X Control R" and "Control V" and "ESC V".

Lesson 4: Searching

For this lesson, you will be reading from your terminal rather than from this paper. Get into Mince as usual, and type:

Control X Control R lesson4.doc

followed by a carriage-return. You should see this text appear on your screen. If it does not, and Mince says "New File" in the echo line at the bottom of the screen, first make sure that you had typed the name correctly. If not, just try again; if so, stop and get help finding the file.

We will be playing around with some new commands on this text. In order to keep reading from your terminal, you will have to type Control V's (View next page) every time you reach the bottom of the scren. Do that now. Type a "Control V" and forget about the paper copy of this lesson.

The first command we will learn this time is the "search" command, "Control S". Suppose someone has given you a draft for corrections, and somewhere in the middle has marked a phrase to be changed. It's easy enough to see it on paper, because the place is marked in red ink or the like. But it requires extra work to find it on the computer. It's almost as though you had been handed the same report, unmarked, and been told that some phrase should be deleted!

Certainly we could scan through the file looking for the marked place using Control V's, but it might take a while. Instead, we can use the computer to search through the file, looking for the proper place.

Try that now. Type:

Control S

(You might want to look back at the paper copy of this lesson for just a little while now, since Control V's won't work while we're in the middle of a search.) Mince will respond to the Control S by printing "Forward Search <ESC>:" in the echo line at the bottom of the screen. Now type in the word:

search

You will see it echoed at the bottom of the screen. If you make a mistake while typing this, you can use the DELETE key, just as you would normally. After you've entered the string to search for, type the ESCAPE key. This tells Mince to begin searching.

A forward search begins at the character the cursor is on, continues until the searched-for item is found, and when it is, positions the cursor just after it. So, when you hit the ESCAPE key, you should have seen the cursor jump to one of the occurrences of the word "search" in this lesson. It should still be there now, unless you've had to type Control V's to read more. If so, try it again. Type:

Control S search ESCAPE

Now just try typing:

Control S ESCAPE

That is, do not type in any character string to be search for; just hit the ESCAPE key right after the Control S. You should notice the cursor jump to the next occurrence of the word "search". When you do not give the Control S command a new character string to look for, it searches for whatever you told it last time. This can be quite a timesaver, especially when working with technical papers, where long, hard-to-type phrases are commonly used several times, and you want to find, say, the sixth occurrence.

What happens if there is no character string exactly like the one you type for Control S? Try it and see. Do a search for a nonsense word. If there is no such word between the cursor and the end of the text, as you see, Mince will cause the terminal to beep and will print the phrase "Not Found" at the right of the echo line.

As you may have guessed by the message that Mince printed in the echo line earlier, "Forward Search", there is a "backward" search as well. It is used to search backwards in the text from the current cursor position. The command is called "Reverse search", and is performed by Control R. All of the features of Control R are identical to Control S, except that, since it searches through the text in reverse, it leaves the cursor before the closest item that matches the string, rather than after it. Notice that the character string to be searched for is saved and is the same one for both Control R and Control S. So, you may find yourself typing "Control S", a string, and the ESCAPE, and then, if you hear a beep, typing "Control R ESCAPE" because you know the string is in the text somewhere.

Play around with these a little bit, then type Control X Control C to leave the editor. The next lesson, Lesson 5, should be read off the printed page, but for Lesson 6 we'll be working exclusively on the terminal again.

Lesson 5: Keyboard Culture

This lesson is a short question-and-answer session. It explains some useful details about the other Mince documentation, as well as some typing conventions used through the rest of this document, the Mince User's Manual, and in Mince's screen display.

(Q) I looked at the user's manual and noticed that Control commands are spelled "C-". Why?

(A) There are three common ways of representing control characters:

(1) Control P
(2) C-P
(3) ^P

The reason for using #2 is that it is shorter to type and easier to discern from surrounding text than #1. That is the format which will be used throughout the rest of these lessons. It is also the format which is used in the Mince User's Manual. Format #3 is used in Mince screen displays. If you had a C-P (Control P, remember) in your text, and representation #2 were used, there would be no way to tell it from the three characters C, -, and P all in a row. The "^" (caret or uparrow) character is much less used, so it is a better representation for use with general text; it stands out almost as much as "C-" and is not quite so likely to be in any text you encounter.

(Q) I thought all control characters I typed were interpreted as commands to control Mince. How on earth could I get one into my text?

(A) With the C-Q command. C-Q stands for "Quote next character". So, typing C-Q C-P would put a C-P into your text buffer. (It would display on the screen window as "^P", remember.) Exactly why you would want to insert control characters is another good question; we don't have any answer for that!

(Q) In the User's Manual, the Escape commands are abbreviated "M-". Is that for the same reason as "C-"?

(A) Basically, yes. The reason for "M-B" being the same as "ESC B" is just a bit more obscure. Once upon a time, someone thought that it would be neat to have yet another Control key, which also worked just like the shift key. They went ahead and made a keyboard which had one; it was called the "Meta" key. (One major terminal manufacturer even offers such a key today.) Programmers who see it think it's "neat"; it's VERY untypewriterlike, though. So, those of us without a Meta key for Meta-commands simulate it by typing an Escape in front of the command character. From now on in these lessons (and in the other manuals) we will use "M-" to be the abbreviation for prefixing a command with the ESC character. So:

```
C-K    is Control K
M-K    is Meta-K, typed ESC K
M-C-K  is Meta-Control-K (!) and
       is typed ESC Control K.
```

These conventions will make it easier for you to read through the text of lessons and manuals more quickly. You will also notice that if you type the Escape key and wait a second, you will see "Meta:" appear in the echo line. This occurs so that you don't forget that you have typed that prefix character.

(Q) Suppose I type C-U and then decide that I really didn't want to repeat any command. What should I do?

(A) Type a C-G. The C-G command flushes all prefix characters. It works just as well with Escape (when you decide you don't want to do a Meta command), C-X, and C-S/C-R (where it cancels the type-in of a search string). C-G makes your terminal bell beep; this is your assurance that the command has worked. Typing C-G just makes the terminal beep otherwise. This could be useful if you give Mince a series of advanced commands that will take a while. You can just type a C-G after all of them, and Mince will beep when it's all done.

(Q) That's interesting. You mean I can type things without waiting for the others to happen? I hate waiting for the screen to redisplay the window!

(A) Absolutely. This speeds typing up for you; it's called "type-ahead". For example, if you know that after the next occurrence of a particular word you want to enter a comma, you can type C-S, the word, an ESC, and a comma immediately thereafter. You don't need to wait for the cursor to jump or the screen to redisplay if you don't want to. Just make sure that what you've done is correct, though. If you type a

C-V immediately after that, you might never get to see the screenful of text after it had been modified. Mince assumes that if you are typing very quickly, you know what you are doing, and would only be annoyed by the flicker and flash of a constantly changing screen. If you've already gone somewhere else by the time the current screenful of text is changed, Mince won't bother with all the flashing. It just always tries to show the screenful of text where the cursor is positioned.

(Q) Great. Then why was there that big explanation of the "Swapping..." message in Lesson 4?

(A) *sigh* That's the one exception to the rule. When Mince uses the disc drives, both to read or write files and to read or write pages of a file, it ignores the keyboard. That is an unfortunate result of using this particular sort of small computer.

(Q) Whenever I want to insert text, I move to the right place in the text and start typing, right? But it's really annoying, when I want to insert whole sentences or paragraphs, to see the line where I'm entering keep moving text over to make room. What do you do about that?

(A) One possible method would be to insert a carriage-return before you start entering text. That way, you will always be typing text on a blank line, and the redisplay won't occur. Unfortunately, if you insert a carriage-return, then start typing, you will still be on the same line you wanted to avoid in the first place! You must insert the carriage-return, then type a C-B to get back to where tha blank line is. There is a command to do this automatically; C-O (for "Open lines") will insert the newline and place you on it to give you a clear line for text entry. From then on, you can just use carriage-return, as you normally would.

(Q) Why do C-S and C-R and the C-X commands put the termination character abbreviations in <...>'s?

(A) Most things Mince displays in angle brackets are specially labelled keys on your keyboard.

<CR>	the Carriage-return key
<ESC>	the Escape key
<DEL>	the Delete key
<LF>	the "Linefeed" key

We also have one called <NL> which you may run across. It stands for "New Line". There is no such key on the keyboard. But if you want to search for one word at the end of a line

followed by another word at the beginning of the next line, you would type C-S, the first word, a carriage-return, the second word, and then the Escape key. In the echo line, you will see "<NL>" where you had typed the carriage-return, to tell you that the search will be looking for a new line inbetween the two words. Just as with typing regular text in the window, the Delete key will erase newlines, just like any other character.

(Q) Thank you for answering EVERY question I could EVER CONCEIVABLY ask about Mince.

(A) Obviously, we haven't even scratched the surface here. If you have questions or forget what certain commands do, try looking in the Mince User's Manual first. If there are no good answers there, give us a call or drop us a note in the mail.

Lesson 6: Killing and Moving Text

During Lesson 6, you should be reading from your terminal again, rather than from this manual. Type to the operating system:

```
mince lesson6.doc
```

and wait for Mince to get started. By putting a file name on the command line, you cause the file to be automatically read in when Mince starts up. As with Lesson 4, if Mince says "New File", and this text does not appear, something is wrong. Determine the cause and use the C-X C-R command to read in the file, if you need to.

In this lesson, we are going to learn how to delete whole bunches of text, rather than characters or words at a time. The command to delete lines is C-K. (Control K stands for "Kill line".) Move the cursor to the first line of this paragraph with C-P or C-N. Now type a C-K. Whatever was on that line has disappeared. Now type another C-K. The line itself has disappeared and all the other lines of text have been moved up.

Now move the cursor to the first line of this paragraph, and move halfway into the line with C-F's or M-F's. Again, type a C-K. You will notice that only the part of the line to the right of the cursor was killed. C-K kills text starting with the character which the cursor is resting on until it reaches a newline, or the newline itself, if the line is blank.

You see that in order to physically remove an entire line AND its contents, you must type two C-K's. Similarly, C-U C-K removes two entire lines and their contents, rather than four. You will find this command useful for retyping previously existing lines of text, where you want to remove what the line (or the "rest" of the line, starting at the cursor) says, but want to fill it with something else.

Typing many C-K's in a row could get to be a drag, especially if you intend to delete an entire paragraph or chapter. You could count the lines to be deleted and type C-U, twice the number, C-K; this would still be pretty time-consuming. (Counting lines is even more annoying than typing C-K many times!) Instead, Mince has a command to kill an entire region of text. It is called C-W (for "Wipe region").

You must define the region of text to be killed first. One end of the region which C-W will wipe out is shown by where the cursor is in the text. The other end of the region is given by a "mark", which we will now learn how to set. The command which sets the invisible mark is C-@ (for "set the mark AT this position"). Move the cursor to the beginning of some line in the middle of this paragraph. Type C-@. (Note that on some terminals you may have to hold down the Shift key and the Control key to get a C-@.) Down in the echo line you should see the message "Mark Set." If it does not appear, you may have a poor terminal whose C-@ doesn't send the command. Try typing C-@ again, and if it doesn't work this time, type a Control SPACE now, and use it from now on if it works, rather than the Control at-sign. If the "Mark Set" message still hasn't appeared down in the echo line, you will have to type ESC SPACE instead, and use the M-<SPACE> command from now on instead of either C-@ or C-<SPACE>.

Having figured that out, move to the end of the same line and type a C-W. Look at the text; the portion of it between the cursor (at the end of the line) and the invisible mark (at the beginning of the line) should have disappeared.

Let's wipe something slightly bigger than a line. Move the cursor to the beginning of this paragraph, and set the mark there by typing C-@. Now move the cursor to the end of this paragraph and type a C-W. You shouldn't be able to read this sentence any more!

What if we make a mistake and wipe out a huge block of text unintentionally? There is a command to retrieve the text which was just killed, C-Y (for "Yank back killed text"). Do that now: type a C-Y and the paragraph we just deleted should appear back where the cursor is now.

Now move the cursor down a few lines and type C-Y again. Another copy of that text appears at the cursor. Type the C-Y again; the text should be replicated one more time. You've just learned how to copy or move text, all with one command, C-Y! To make a copy of some text, wipe it out (using C-@ and C-W) and immediately yank it back with a C-Y in that spot (i.e., without moving the cursor). Then, move the cursor to the spot where you would like the copy to appear, and type another C-Y. If you want to move a block of text rather than copy it, just don't type the C-Y at the original position. (If deleting text and then yanking it back in the same place in order to do a copy of some text makes you nervous, you might want to look at the description of the M-W command in the Mince User's Manual. It works similarly to C-W, but it does the first C-Y automatically for you.)

C-W is not the only command which saves text in case you want to yank it back. C-K, M-D, and M-<DEL> all save text as well. In general, if you delete anything larger than a

character, Mince will save it for you in case you want to move or copy it (or undo a mistake!).

As an example, move the cursor to the beginning of this paragraph, and type a couple of C-K's. Notice that a "plus" sign has come on at the right edge of the mode line. This means that if you continue to type C-K's, the text killed will be added on to whatever killed text is already being stored. So, you can kill a region of text, with either a C-@/C-W or with a bunch of C-K's (or even M-D's or some combination of all of these), and a C-Y will still yank the entire region back.

Mince will only store your "most recent" bunch of text kills, however. What determines what is "recent" and what is not? The plus-sign at the right of the mode line. If you are about to delete something larger than a character and the plus is not there, you will be throwing away whatever previous bunch of kills you did. In general, all this amounts to is that, if you give any movement commands or insert anything after killing some text, you will "close off" the current group of kills. (You will also see the plus-sign go away.) Any C-Y's you type after this will retrieve that group. Any C-K's, M-D's, M-<DEL>'s, or C-W's you do after this will throw away that group of kills and start a brand new one.

There is a command to "turn on the plus sign". This is used if you want to move groups of lines from several different places all to one place. Certainly you could do this manually, doing a few C-K's, moving to the right place, doing a C-Y, going somewhere else and doing some C-K's or a C-W, moving back to the right place, doing another C-Y, etc. It would be much easier to do C-K's or C-W's in all the various places and then yank the whole thing back with a single C-Y. But we said earlier that movement away from the place of the text killing or wiping causes the plus-sign to go off and the current bunch of kills to be "closed off". The command to turn the plus sign back on is M-C-W. (This is your first "M-C-" command. Remember that you just type ESC Control W.) You can remember this command by its intimate relation with C-W (and M-W, if you use it).

As M-C-W is sort of hard to think about, there is no substitute for practice. Move to the beginning of this paragraph and kill the text on the first line, in whatever manner is convenient. Then move to the last line. Notice that the plus in the mode line has gone off. Type a M-C-W to turn it on, then kill off the text on the last line. Now move into the middle of what's left and type a C-Y. You will see both the first and last lines appear there at once.

Play around with all these commands a little bit. Try deleting the last five words of this sentence and yanking them back at the beginning of the sentence. Then, move this paragraph so that it is the first paragraph in the lesson. Try doing C-W's with the cursor both before and after the invisible mark you set,

and observe the results. Killing and yanking text is one of the most complicated, confusing, and hard-to-explain features of Mince. However, once you do understand it, it's also one of the most useful and convenient features. Take your time and experiment enough to make sure that you understand these commands.

When you're all done type a C-X C-C and answer "Y" for "yes" when Mince asks you if you want to abandon the text buffer without saving it. You wouldn't want to destroy the file "lesson6.doc" for the next person, after all. We'll learn more about text buffers and neat tricks you can do with C-Y and lots of different buffers in Lesson 8.

Lesson 7: Text Processing Commands

This lesson is about commands which operate upon words, sentences, and paragraphs. You may remember that in Lesson 2 we made an analogy between the "C-" commands and the "M-" commands (although we didn't call them that back then) and thereby learned M-F, M-B, M-D, and M-<DEL> to operate on words. We also learned M-A and M-E for moving by sentences and M-< and M-> for moving to the beginning and end of the entire buffer of text.

In this lesson you should have some text to play around with, and it should have a few paragraphs. The text of this lesson is an ideal example, but you may want to use some text that you have been working on and are familiar with. Read the file into Mince with C-X C-R. (If you decide you want to play with this lesson text, read in "lesson7.doc".)

The first three commands we will learn are M-U, M-L, and M-C. All three work on words. M-U stands for "Uppercase word". M-L stands for "Lowercase word". M-C stands for "Capitalize word". Try each of these on some words.

You may notice one very strange thing about these commands -- they do not actually look for a word in order to begin recasing letters. They merely look for the first alphabetic or numeric character starting at the cursor. For example, if you had the word "Macpherson" and positioned the cursor on the "p", a M-C would have produced "MacPherson", a M-L would have left the word unchanged, and a M-U would have created "MacPHERSON". In any case, the word casing commands would have left the cursor on the space following the word. Although these commands do not look backwards for the beginning of a word to work on if they are in the middle of one, they WILL look forward as much as necessary. (You may remember something similar from the discussion on M-F and M-D in Lesson 2.) Thus, C-U 5 M-U will uppercase the 5 following words, regardless of whether they are separated by spaces, newlines, punctuation, or other special characters.

The next command we will learn deals with sentences. It is M-K, the "Kill sentence" command. M-K is similar to C-K; it saves what it kills in case you want to do a C-Y later. Also, just as a C-K typed in the middle of a line only kills forward to the end of the line, a M-K, if typed in the middle of a sentence, will only kill from that point to the end of the sentence. (If

you are in the middle of a sentence and want to kill it all, beginning to end, type M-A (beginning of sentence, learned in Lesson 2) then M-K.)

M-K can be fooled by abbreviations because they have periods in them and hence look just like ends of sentences. But better too little deleted than too much. If M-K ever stops before you want it to, just type it again, and the "rest" of the sentence will disappear. Try a few M-K's and then a C-Y to restore it all. You may notice that the last M-K will not delete the two spaces after sentence punctuation. It truly only deletes from where the cursor is to the next end-of-sentence which follows it. You may have to clean up the extra spaces manually.

The rest of the commands we will learn about in this lesson deal with entire paragraphs of text. The two simplest commands are M-[and M-]. They move to the beginning and end, respectively, of whatever paragraph you are in. (You can remember these commands because they look very similar to M-< and M->, which we learned in Lesson 2.) If you are inbetween two paragraphs and not really "in" either one, M-[will move to the beginning of the preceding paragraph and M-] will move to the end of the following one.

Try these a few times, at various places in the text. You may notice the cursor stop in places you didn't think were paragraphs, for example in the middle of lists or in front of indented examples. So, what makes a paragraph? As far as these and all other Mince paragraph commands are concerned, a paragraph begins (or ends, if you prefer to think of it that way) with:

(1) A blank line. (that is, two newline characters in a row.)

(2) A line started by hitting the TAB key. (That is, a newline character followed by a tab character.) Note that this is NOT the same as a line with leading spaces.

(3) A line started with a commercial at-sign ("@"). This is for compatibility with a particular set of "text-formatter" commands; chances are rare that you will type a line with an atsign character as its first character in ordinary text.

Another command for dealing with paragraphs of text is M-Q. M-Q "fills" entire paragraphs of text, rearranging words and lines so that the right-hand margin is consistent. (We couldn't make M-Q stand for anything mnemonic; you'll just have to remember it by rote.) This allows you to type paragraphs as carelessly as you like with regard to right margin, and at the end of a typed paragraph, with a single keystroke go back and tidy up the text. It also allows you to keep the text neat in the same way: When modifying a peviously existing paragraph you may add or delete words with no regard for existing margins, because M-Q can fix them up when you're done.

Try filling a few paragraphs. (To make M-Q work on a particular paragraph, position the cursor anywhere in it, then type the M-Q.) If M-Q doesn't do anything to the text, then the paragraphs are already as well filled as they can be. Try inserting some extra text into a line and doing another M-Q. If you notice M-Q joining any paragraphs together, this is because they were not separated properly. M-Q uses the same paragraphing rules as M-[and M-] do. If you want to make certain of how much text you are about to fill with M-Q, you can check to see where the edges are by tpying M-[, then M-], then (if you're satisfied that the boundaries are correct) M-Q.

Of course, no harm is done if M-Q joins two groups of text which you desired as separate paragraphs. You can easily position yourself to where you want the new paragraph to occur and insert the proper separator (either a blank line or a leading tab). Then just M-Q the second new paragraph. Similarly, to join and refill two paragraphs, merely delete the separator characters and use M-Q. This can be particularly useful for modifying memoranda, manuals, or legal documents, where text is frequently repositioned to change paragraph structure and coherence.

You may have wondered how M-Q knows where the right margin is supposed to be. There is a default margin column, which you can set yourself if you choose. The command to do this is "C-X F". That is, type a C-X, and then an "F". (This command stands for "Fill column".)

There are two ways to use C-X F. One is with a universal repeat. Type:

C-U 70 C-X F

and you will see a message in the echo line saying "Fill Column = 70". Try setting the fill column to something between 75 and 80 and do a M-Q. The other way to set the fill column is "by eye". Move the cursor to somewhere in the middle of a line, and type C-X F without an repeat count. You will see a new fill column setting appear in the echo line. Type a M-Q again and notice where the new right margin is. If you give C-X F an repeat with C-U, it will set the fill column to that number. If you do not, it will set the fill column to wherever the cursor is at the time.

Another useful command is "C-X .". (That is, a Control X, followed by a period.) This command sets the paragraph indentation column, and is used to make an entire paragraph be indented away from the left edge of the screen. This command is analogous to setting the left margin on a typewriter, but this margin is used only by the text-filling commands. Type:

C-U 10 C-X .

and you will see the message "Indent Column = 10" appear in the echo line. Type a M-Q and look at the results.

The "C-X ." command is useful for making narrower paragraphs, perhaps for example text or quotes. To make one, make the indent column larger and the fill column smaller, then type the text and fill it with M-Q. Then, return the fill and indent columns to their original positions. (The standard settings are usually gotten by "C-U 0 C-X ." and "C-U 65 C-X F".)

If you find typing M-Q repeatedly while inserting text annoying after a while, you may be interested in "Fill Mode". This causes carriage-returns to happen automatically while you are typing, so that not only do you not have to worry about the right margins, but you may never have to type M-Q unless you go back to repair old text. For more information look in the Mince User's Manual under "modes".

Lesson 8: Buffers

You may remember the word "buffer" from Lesson 3, when we learned how to read and write files from the text buffer. As you recall, we said that the buffer was a place for storing the text while we manipulated it, and that files were copied into it and replaced from it.

Mince has more than one buffer for manipulating text. This can be handy when you are working on several files at once. For example, this manual is made up of several chapters, and if we wanted to edit two or three of them, making changes to one based upon the others, it would be nice to be able to read them all into Mince at the same time. Each file we chose to use could be read into one of Mince's buffers.

For example, start using Mince to edit this chapter, called "lesson8.doc", by typing:

C-X C-R lesson8.doc <CR>

to Mince. (Remember that the disc drives are labelled, and that if the file can't be found, you may have to type "B:lesson8.doc" or some such for the file name.) (Remember also that "<CR>" stands for "carriage-return. If you should forget what character to terminate the filename with, just look in the message asking what file to read; it will always have the "<CR>" in it for file commands.)

Look at the mode line at the bottom of the screen. Notice the portion that says "main: lesson8.doc". In the mode line, "main" is the buffer name. Since we have multiple buffers for storing text, they must be named in some way, just like files or disc drives. Buffer names may be from one to eight characters long. "main" is the one you get automatically when you start Mince up. Each of these buffers of text also has a filename associated with it. In this case, the filename is "lesson8.doc". We talked about filenames in Lesson 3.

There is a command which will list on the screen what buffers of text currently exist. This command is C-X C-B, the "list Buffers" command. Try it now. Type:

C-X C-B

You should see a display at the top of the screen, overwriting whatever text was there before. The text is not gone, just momentarily not displayed; this is an exception to the rule that what you see on the screen is what is in your document. Take a quick look at the display line with the buffer list on it. It has the name, "main", and the file name, "lesson8.doc", and a number, which tells you how many characters are in the buffer of text. Type a C-L now to redisplay the screen. The buffer list has been replaced by the original text again.

There is a command to create a new buffer, the C-X B command. It stands for "goto Buffer", and looks similar to the C-X C-B command, so you can remember them both fairly easily. Try this command now. Type:

C-X B

Mince will ask you for the name of a buffer to use, by displaying a message in the echo line. Type in the name "other", followed by a carriage-return. Mince will ask you if you want to create a new text buffer. We do, so type a "Y" to answer yes. Now the screen is devoid of text. Look at the mode line, and notice that the buffer name is now "other". We moved into a brand new text buffer, which has no characters in it. Notice also that the filename in the mode line associated with this buffer is called "DELETE.ME" This is so that if you mistakenly type a C-X C-W command to write the file without giving a filename, the text will be stored in a conspicuously-named file. Type a few characters just so the buffer isn't empty.

Now type a C-X C-B again to see a new list of the text buffers. Notice that the list now shows two buffers, "main" and "other". Going to a new text buffer did not delete the old one; it is just waiting for whenever you want to go back to it. Do that now; type:

C-X B main <CR>

The Lesson 8 text will appear again. Note that C-X B didn't ask you if you wanted to create a new buffer, because one by that name already existed. Type the C-X C-B command again, to list the buffers. Notice that there is an asterisk beside the "other" buffer. This means the same thing as the star on the mode line does: the text buffer hasn't been written out to a file since it has been modified.

Now go to another new buffer, called "lesson6". Type:

C-X B lesson6 <CR> Y

(The "Y" is in answer to the question asking if we want to create a new buffer.) Now that you're in the "lesson6" buffer, read in the file containing Lesson 6. Type:

C-X C-R lesson6.doc <CR>

The text will appear (if you got the file name right) and the mode line will now have a section saying "lesson6: LESSON6.DOC". Type a C-X C-B again to get your bearings, if you like.

It is usually very useful to have the buffer name be the same as the first half of the file name which Mince and the computer system use. We just accomplished that by creating a buffer with a name appropriate to the file we were about to read in. But Mince can do this automatically for us. The command C-X C-F (for "Find File") will read in a file, in the same manner as C-X C-R, but will automatically create a buffer of the appropriate name for it. Try it now. Type:

C-X C-F lesson4.doc <CR>

The mode line will now say "lesson4: LESSON4.DOC", and the text of Lesson 4 should be on the screen.

The C-X C-F find-file command does just a little more than automatically selecting a "nice" buffer name. It will look through all the Mince buffers you have to see if the file you want to find has already been read into a text buffer before. If so, it just switches to that buffer, rather than creating a new one and reading in the file. This is almost certainly what you want; if you had made changes to a buffer containing a file and then did a find-file, you would want to see the modified version. Try it now. Type:

C-X C-F lesson6.doc <CR>

Note that this puts you back in the "lesson6" buffer. Try:

C-X C-F delete.me <CR>

This puts you into the buffer called "other", with its original file name. So, C-X C-F always does an automatic C-X B command for you, either to a buffer which contains the file name you want, or to a brand new buffer into which it reads the file. It effectively prevents you from ever having to remember whether of not you had read in a file. Just use C-X C-F all the time.

Do a C-X C-B to get a handle on what text buffers and files we have again. There is certainly a lot of junk, so let's get rid of some of those old buffers. The command to do this is C-X K (for "Kill buffer"). Type:

C-X K lesson4 <CR>

This will kill the buffer called "lesson4", which contains the file "lesson4.doc". Be sure to keep the two distinct in your mind. C-X K and C-X B work with buffer names, and C-X C-F, C-X C-R, and C-X C-W all work with file names. Do a C-X C-B again,

and notice that the buffer named "lesson4" is no longer there.

Now type:

C-X K other <CR>

This command is intended to delete the buffer called "other", which just happens to be the one on your screen now. Mince will not delete a buffer which we are currently working on, and so it asks us which buffer we would like to go to instead. Type "main" followed by a carriage- return, and Mince will then switch you back to the "main" buffer (which has the Lesson 8 text in it), and try to delete the "other" buffer.

But there is another message at the bottom of the screen: Mince will ask you if you are sure that you want to kill the buffer, because it has some text (those few characters) in it which has not been written out to a since the changes were made. (Remember the star in the C-X C-B listing and at the right end of the mode line now?) Answer "Y" for yes, and let Mince delete the buffer. You can check this out with a C-X C-B listing if you like.

What advantages does using several buffers have besides allowing you to look at several files back and forth? It allows you to move or copy text from one to the other as well. In Lesson 6, we used the killing mechanism (or C-W wiping or M-W copying mechanism) to move or copy text from one place to another in a file. This method works on multiple files in separate buffers as well. You can kill some text in one buffer, do a C-X B command to another buffer, and yank back the killed text in the new buffer.

Let's work through an entire example in detail. This is a chance for us to review some of the many commands that you have learned in Lessons 6, 7, and 8. The task is to take the first paragraph of this text, Lesson 8, and make it the first paragraph of Lesson 6. The buffer which we are now in, "main", contains the file "lesson8.doc". The other buffer, "lesson6", contains the file "lesson6.doc".

(1) M-< to the beginning of lesson8.

(2) C-N down to the first line of the paragraph.

(3) C-@ to set the mark at the beginning of the first line.

(4) M-] to get to the end of the paragraph.

(5) C-W to wipe out the paragraph.

(6) C-X B to the buffer "lesson6"

(7) M-< to the beginning of that buffer.

(8) C-N down to just before the start of the first paragraph.

(9) C-Y to yank the text back. It should now be the first paragraph of Lesson 6.

Don't forget that the text is still yankable again; you may want to go back to the other buffer and repair it with a C-Y.

We have learned some commands in this lesson which you might not use quite at first. Sooner or later, though, you will be using them regularly. If you get into the habit of using C-X C-F to read in a file initially, you will find it easier to use the multiple buffers later when you need to. Experiment some more with buffers and moving text back and forth among them. When you exit Mince with the C-X C-C command, be sure NOT to write out the buffers, so that Lessons 6 and 8 are intact for the next person to use them.

Epilogue

This document is merely an introduction to the features of Mince. Further information on commands ranging from the obscure (e.g. C-T, a command to transpose characters) to the sublime (e.g. M-C-R, the Query Replace command, which replaces occurrences of one string with another, allowing you to view results, and either accept or reject them) is available in the Mince User's Manual. Other useful features are available, such as "fill mode" or "page mode", which allow certain commands to do special things when inserting or deleting text.

Having practiced all the commands used in these lessons, you should be able to perform almost all duties required of you. There are a few commands left to learn, however, which can make the job of editing a little easier. Read the user's manual when you get a chance! Best wishes!

You are looking at the Mince tutorial. While you can read this in hardcopy form, it is more helpful to sit down at a video terminal and use Mince to read it. Just type the command "mince prgintro.doc".

Mince commands are generally shifted by the CONTROL key (sometimes labelled CTRL or CTL) or prefixed by the ESCAPE key (sometimes labelled ESC or ALT). Rather than write out ESCAPE or CONTROL each time we want you to prefix a character, we'll use the following abbreviations:

C-<char>	means hold the CONTROL key down and type a character.
M-<char>	means type the ESCAPE key, release it, then type the character. (The "M" stands for "Meta-command"; ESC is a substitute for another type of shift key, the imaginary "meta-shift" key.)

Thus, C-F would be hold the control key and type F. You will often be asked to type characters to see how they work; don't actually do this, however, until you see >> at the left of the screen. For instance:

>> Now type C-V (View next screen) to move to the next screen. (go ahead, do it by depressing the control key and V together). From now on, you'll be expected to do this whenever you finish reading the screen.

Note that there is an overlap when going from screen to screen; this provides some continuity when moving through the file.

The first thing that you need to know is how to move around from place to place in the file. You already know how to move forward a screen, with C-V. To move backwards a screen, type M-V (depress and release the <ESC> key, then type V.)

>> Try typing M-V and then C-V to move back and forth a few times.

SUMMARY

The following commands are useful for viewing screenfuls:

C-V	Move forward one screenful
M-V	Move backward one screenful
C-L	'Refresh' the current screen.

>> Try C-L now. (You'll notice that it centers the screen where the cursor currently is.)

BASIC CURSOR CONTROL

Getting from screenful to screenful is useful, but how do you reposition yourself within a given screen to a specific place? There are several ways you can do this. One way (not the best, but the most basic) is to use the commands Previous, Backward, Forward and Next. As you can imagine, these commands (which are given to Mince as C-P, C-B, C-F, and C-N respectively) move the cursor from where it currently is to a new place in the given direction. Here in a more graphical form are the commands:

```
                    Previous line, C-P
                            :
                            :
Backward, C-B .... Current cursor position .... Forward, C-F
                            :
                            :
                    Next line, C-N
```

You'll probably find it easy to think of these by letter. P for previous, N for next, B for backward and F for forward. These are the basic cursor positioning commands and you'll be using them ALL the time so it would be of great benefit if you learn them now.

>> Try doing a few C-N's to bring the cursor down to this line. Move into the line with C-F's and up with C-P's. Now use these four commands to play around a little. Try moving off the top of this screen and see what happens.

When you go off the top or bottom of the screen, the text beyond the edge is shifted onto the screen so that your instructions can be carried out while keeping the cursor on the screen.

>> Try to C-B at the beginning of a line. Do a few more C-B's. Then do C-F's back to the end of the line and beyond.

If moving by characters is too slow, you can move by words. M-F (remember, type <ESC> F) moves forward a word and M-B moves back a word.

>> Type a few M-F's and M-B's. Intersperse them with C-F's and C-B's.

You will notice the parallel between C-F and C-B on the one hand, and M-F and M-B on the other hand. Very often Meta characters are used for operations related to English text whereas Control characters operate on the basic textual units that are independent of what you are editing (characters, lines, etc). There is a similar parallel between lines and sentences: C-A and C-E move to the beginning or end of a line, and M-A and M-E move to the beginning or end of a sentence.

>> Try a couple of C-A's, and then a couple of C-E's.
Try a couple of M-A's, and then a couple of M-E's.

See how repeated C-A's do nothing, but repeated M-A's keep moving farther. Do you think that this is right?

Here is a summary of simple moving operations including the word and sentence moving commands:

C-F	Move forward a character
C-B	Move backward a character
M-F	Move forward a word
M-B	Move backward a word
C-N	Move to next line
C-P	Move to previous line
C-A	Move to beginning of line
C-E	Move to end of line
M-A	Move back to beginning of sentence
M-E	Move forward to end of sentence
M-<	Go to beginning of file
M->	Go to end of file

>> Try all of these commands now a few times for practice. Since the last two will take you away from this screen, you can come back here with M-V's and C-V's.

These are the most often used commands. Like all other commands in Mince, they can be given arguments which cause them to be executed repeatedly. The way you give a command a repeat count is by typing C-U and then the digits before you type the command. (C-U stands for "Universal argument".) The digits are echoed at the bottom of the screen slowly, just after you type them. Notice that just after you type C-U, the message "Arg: 4" appears there. If no numbers are typed after the C-U, it executes the following command 4 times. For now, though, just type in numbers.

For instance, C-U 8 C-F moves forward eight characters.

>> Try giving a suitable argument to C-N or C-P to come as close as you can to this line in one jump.

INSERTING AND DELETING

If you want to type text, just do it. Characters which you can see, such as A, 7, *, etc. are taken by Mince as text and inserted immediately. You can delete the last character you typed by doing <DEL> (sometimes labelled "DELETE" or "DEL" or even "RUBOUT"). More generally, <DEL> will delete the character immediately before the current cursor position.

>> Do this now, type a few characters and then delete them by typing <DEL> a few times. Don't worry about this file being changed; you won't affect the master tutorial, because this is just a copy of it in your Mince editing buffer.

Notice that a "*" appeared in the line at the bottom of the screen. This means that the text on your screen is different that the text you read in, and hasn't been written out to a file.

Remember that most Mince commands can be given a repeat count; Note that this includes characters which insert themselves.

>> Try that now -- type C-U 8 * and see what happens.

You've now learned the most basic way of typing something in Mince and correcting errors. You can delete by words or lines just as you can move by words or lines. Here are some of the delete operations:

<DEL>	delete the character just before the cursor
C-D	delete the character that the cursor is positioned on
M-<DEL>	delete the word before the cursor
M-D	delete the word after the cursor
C-K	delete (kill) from the cursor position to the end of line
M-K	delete (kill) to the end of the current sentence

Notice that <DEL> and C-D vs M-<DEL> and M-D extend the parallel started by C-F and M-F. C-K and C-E are similar to M-K and M-E.

Now suppose you delete something, and then you decide that you want to get it back? Well, whenever you delete something bigger than a character, Mince saves it for you. To yank it back, use C-Y. Note that you don't have to be in the same place to do C-Y; this is a good way to move text around. Generally, the commands that can destroy a lot of text will save it, while the ones that attack only one character, or nothing but blank lines and spaces, will not save them.

For instance, type C-N a couple times to position the cursor at some line on this screen.

>> Do this now, move the cursor and kill that line with C-K.

Note that a single C-K will kill the contents of the line, and a second C-K will delete the line itself, and make all the other lines move up. The text that has just disappeared is saved so that you can retrieve it. To retrieve the last killed text and put it where the cursor currently is, type C-Y.

>> Try it; type C-Y to yank the text back.

Think of C-Y as if you were yanking something back that someone took away from you. Notice that if you do several C-K's in a row the text that is killed is all saved together so that one C-Y will yank all of the lines. A way to tell if this is going to happen or not is the "+" which will appear on the line at the bottom of the screen. If it is present, whatever text is killed will be appended to whatever is already there.

>> Do this now, type C-K several times.

Now to retrieve that killed text:

>> Type C-Y. Then move the cursor down a few lines and type C-Y again. You now see how to copy some text.

FILES

In order to make the text you edit permanent, you must put it in a file. You put your editing in a file by writing or saving the file. If you look near the botton of the screen you will see a line that starts with "Mince Version 2.3 (Normal) prgintro:" and continues with the filename PRGINTRO.DOC. This is the name of the permanent file in which the Mince tutorial is stored. This is the file you are now editing. Whatever file you edit, its name will appear in the same spot.

The commands for reading and saving files are unlike the other commands you have learned in that they consist of two control characters. They both start with the character Control-X. There is a whole series of commands that start with Control-X; many of them have to do with files, buffers, and related things, and all of them consist of Control-X followed by some other character.

C-X C-W	writes out the editing buffer
C-X C-R	reads a file into the editing buffer

In addition, each of these commands asks for a filename to use. Enter the name, and finish it by typing a carriage-return (<CR>).

>> Go ahead and try that now; type C-X C-W, and when Mince asks for a filename, type GARBAGE.TXT, then type a <CR>. Note that the mode line has now changed to reflect the new file name. (Don't forget to ERAse the file after you're done sometime.)

If you forget to write out your work and try to read another file, Mince will remind you that you made changes and ask you whether to clobber them. (If you don't save them, they will be thrown away. That might be what you want!) You should answer with a "N" to keep your editing buffer intact or a "Y" to clobber it and read in the new file anyway.

To make a new file, just C-X C-R it "as if" it already existed. Mince will echo "New File" at the bottom of the screen. Then start typing in the text. When you ask to write the file, Mince will really create the file with the text that you have inserted. From then on, you can consider yourself to be editing an already existing file.

Another command is available to prevent retyping filenames all the time.

C-X C-S saves the file

This command just rewrites the editing buffer to whatever file name is in the mode line at the bottom of the screen. It may save some typing.

It is not easy for you to try out making a file and continue with the tutorial. But you can always come back into the tutorial by starting it over and skipping forward. So, when you feel ready, you should try reading a file named "FOO", putting some text in it, and writing it; then exit from Mince and look at the file to be sure that it worked.

One more immediately useful command is C-X C-C, which tells Mince you'd like to stop editing. (Think of it as an augmented C-C, which usually works in the operating system to get you out of programs.) This does NOT save your file. It will ask if you really want to quit if you have not written out the editing buffer, however.

MODE LINE

If Mince sees that you have typed an <ESC> or C-U or C-Q or a C-X and have not typed the following character in the command sequence, it will show you the prefix you have typed in an area at the bottom of the screen. This line is called the "echo line"; it echoes numbers typed after a C-U, characters to be included in search strings, and some progress information when file I/O is going on. This is just the last line at the bottom.

The line immediately above this is called the MODE LINE. You may notice that it begins

Mince Version 2.3 (Normal) buffer: DRIVE:FILENAME -nn%- *

This is a very useful "information" line. You already know what the filename means -- it is the file you have read. What the -nn%- means is that nn percent of the file is above the cursor. The "*" means that the editing buffer has been changed since the file was last written. You also know what the "+" means in relation to the C-K command.

SEARCHING

Mince can do searches for strings (these are groups of contiguous characters or words) either forward through the file or backward through it. To search for the string means that you are trying to locate it somewhere in the file and have Mince show you where the occurrences of the string exist. The command to start a search is C-S. Down in the echo area, you will notice "Forward Search: <ESC>:" appear. Type in the string you want to search for (which will appear in the echo area also). When you finish, type the ESCAPE key and Mince will try to find the next occurrence of the string in your text.

If no such occurrence exists Mince beeps and tells you that the string was not found. In addition, if you decide you really don't want to search after all, type C-G and Mince will erase the search string and cancel the C-S command. (More generally, C-G cancels any command; for example, if you mistakenly type <ESC> or C-U when you didn't mean to, type C-G to flush the prefixes.) If you are in the middle of an search and type the DELETE key, you'll notice that the last character in the search string is erased.

>> Now type C-S to start a search. Type the word "search" followed by an <ESC>. Notice where the cursor is positioned to. Now type C-S again, immediately followed by <ESC>. Mince will search for whatever it searched for last time if no new string is given.

The C-S starts a search that looks for any occurrence of the search string AFTER the current cursor position. But what if you want to search for something earlier in the text? To do this one should type C-R for Reverse search. Everything that applies to C-S applies to C-R except that the direction of the search is reversed.

GETTING MORE HELP

In this tutorial we have tried to supply just enough information to get you started using Mince. There is so much available in Mince that it would be impossible to explain it all here. However, you may want to learn more about Mince since it has numerous desirable features that you don't know about yet. Documentation is available online and in hardcopy form. The Mince User's Manual completely describes the commands presented in this tutorial, as well as the more sophisticated commands, modes, and editing features.

CONCLUSION

You'll probably find that if you use Mince for a few days you won't be able to give it up. Initially it may give you trouble. But remember that this is the case with any editor, especially one that can do many things.