T-EDIT, A Collection of Editing Macros for TEX

Larry F. Bennett
Department of Mathematics
South Dakota State University
Box 2220
Brookings, South Dakota 57007-1297
Phone: 605-688-6218
Bitnet: MA01@SDSUMUS.BITNET

Abstract

T-EDIT is a powerful collection of editing macros designed specifically for TEX. The macros are designed to be used with the KEDIT editor on IBM PC or PC-compatible computers. The user is aided in every step of the preparation of a completed document. Menus or prompting messages are included with many of the macros. Over 1250 TEX and $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX control sequences may be accessed through menus. The control sequences may either be inserted directly into the text or assigned to function keys. T-EDIT can be used to control the TEXing and possible previewing of output. Debugging features are included. Macros generating several lines of complicated TEX source code are available, and TEX macros have been designed to be used with several of the code-generating macros.

Introduction

T-EDIT is a powerful collection of editing macros, TEX execution control macros, and a collection of special TEX macros to be used in conjunction with the other macros. The macro package is designed to be used with the KEDIT editor on IBM PC or PC-compatible computers, but can probably be modified to be used with other editors in other environments. In addition, although it currently makes use of a PCTEX implementation of TEX, the ArborText previewer, and the MicroSpell spelling checker, T-EDIT is in no way bound to these software packages. With a few simple changes, it should work with other software.

The editing macros are currently all written in the macro language KEXX, which is essentially a subset of the computer language REXX.

The T-EDIT macro package was developed to make the entire process of creating a TEX source program, TEXing it, previewing it, and debugging it into a fairly simple task. This has been accomplished in many instances by using menus and prompting messages within the macros which guide the user through each required step. Like TEX macros, existing T-EDIT macros may be modified if desired, and new T-EDIT macros may be created. In addition, menus may be modified and created.

The present version of T-EDIT was designed under the assumption that its primary use would be for the entry of mathematical text into source files. Consequently, many T-EDIT macros have been created for specific mathematical applications. Furthermore, the macros have all been designed under the assumption that the user will employ Plain TEX and/or $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX. However, since T-EDIT may be easily modified and expanded, it would be a simple matter for a user to modify it to suit his or her needs.

An editing macro package for TEX based upon the KEDIT editor was developed in 1989 by Don L. Riley and Brad L. Halverson (See Riley and Halverson). These macros were based mainly upon REXX macros instead of KEXX macros. T-EDIT was developed independently. Consequently, the two collections of macros are completely different and were created with different needs in mind.

In the remainder of this paper, I shall describe what T-EDIT is capable of doing, although I will *not* be able to discuss everything. In most cases, I will not be able to go into too much detail about how it accomplishes what it does. This is simply because of the enormous number of the macros employed by T-EDIT, the length of some of the macros, and the fact that the macros are written in a computer language which most readers may not be familiar with. This paper should be considered as an *introduction* to T-EDIT.

Editing Features

Special keys. T-Edit employs special keys and key combinations to implement KEXX macros. IBM compatible Personal Computer keyboards are ordinarily supplied with ten so-called *Function* keys, which are designated on the keyboard as `F1`, `F2`, ..., `F10`. In addition, keyboards which are referred to as *extended keyboards* have two additional Function keys: `F11` and `F12`. In this paper, it will be assumed that an extended keyboard is being employed. There is also a *Control* key, which is denoted by `Ctrl`, and there is an *Alternate* key which is abbreviated `Alt`. *Key combinations* may be formed using `Ctrl` and `Alt`. To form a key combination using `Ctrl`, the `Ctrl` key is held down and a second key is pressed. If the second key happens to be the Function key `F5`, then `Ctrl-F5` is used to denote this key combination. Similarly, if `Alt` is held down and `P` is pressed, then this gives the key combination `Alt-P`. Any one of the key combinations `Alt-F1`, `Alt-F2`, ..., `Alt-F12` will be referred to as an `Alt-F` key. The purpose of the `Alt-F` keys will be discussed later. The Escape key, which is denoted by `Esc`, is another special key that is employed in various T-EDIT macros.

KEDIT allows a user to assign a KEXX macro to any key or combination of keys, and the macro is stored in computer memory. Such a macro will be referred to as both a *level-0* macro and as the macro *associated with* the key or combination of keys. KEDIT refers to the associated macro using the name of the key or key combination it is associated with. For example, the macro associated with `Ctrl-T` is named `Ctrl-T` by KEDIT. This naming convention makes it possible to initiate the macro without pressing the corresponding key or combination of keys. For example, including the command `'macro Ctrl-T'` in another KEDIT macro will cause the macro associated with the key combination `Ctrl-T` to be executed.

Throughout this paper, a *simple macro* should be thought of as level-0 macro which does *not* call upon any macro stored on the hard disk. A *redirection macro* is a level-0 macro which calls immediately upon a macro stored on the hard disk. Ordinarily, T-EDIT uses the same name for the macro on the hard disk as the name of the redirection macro. For instance, `Ctrl-T` is a redirection macro, and it calls immediately upon the macro `Ctrl-T.kex` on the hard disk. The extension `kex` is simply the default extension employed by KEDIT for macros stored on the hard disk. Whenever a macro which resides on the hard disk is called upon, it is loaded into memory, executed, and then released from memory. These macros may also call upon other macros stored in memory or on the hard disk. Any macro T-EDIT associates with a key or combination of keys is either a simple macro or a redirection macro. However, T-EDIT makes use of many more redirection macros than simple macros, and throughout the remainder of the paper, all macros associated with keys or key combinations should be assumed to be redirection macros unless specified otherwise.

KEDIT only allows macros to be associated with a single key or a key combination consisting of two keys. In order to create the illusion of *extended key combinations*, T-EDIT makes use of selection macros. By definition, a *selection macro* is a macro which is stored on the hard disk whose *only* purpose is to allow the user to select from other macros which are stored on the hard disk by simply pressing a key or combination of keys.

Although several selection macros are employed by T-EDIT, only three are going to be mentioned in this paper. As mentioned earlier, the redirection macro `Ctrl-T`, associated with the key combination `Ctrl-T`, calls upon the macro `Ctrl-T.kex` which is stored on the hard disk. The macro `Ctrl-T.kex` is a selection macro. Once this macro is initiated, a message appears on the screen informing the user of possible choices of keys to press next. One of the options which is available, and which will be discussed in more detail later, is to press `O`. The character associated with the key pressed is read into a KEDIT variable called `readv.1`. Next, a substitution is made in a KEDIT command which is similar to, but slightly more complicated than the KEDIT command

```
'macro \Ctrl-T'readv.1'.kex'.
```

Thus, the macro called upon at this point is actually `Ctrl-TO.kex`. Although there is no message to the effect that it is possible to enter the extended key combination in any other way, the same results are achieved by pressing `Ctrl-O` instead of `O`. This option is allowed since after pressing `Ctrl-T` it is very easy to forget and press `O` next while still holding down the `Ctrl` key. Because of this additional option, the extended key combination `Ctrl-T` followed by either `O` or `Ctrl-O` will be designated as `Ctrl-TO`. Similar notation will be used to denote all other extended `Ctrl-T` key combinations as well as extended key combinations involving `Ctrl-I` and `Ctrl-D`, which are also associated with redirection macros and which call upon selection macros.

Throughout the remainder of this paper, extended combinations will be used without further explanation.

Automatic key assignments. Whenever a TEX source program is edited with T-EDIT, a great number of macros are automatically associated with keys and key combinations. It is the collection of macros associated with various keys and key combinations which does much of the work for T-EDIT. Some of these macros call upon menus which may be used to select other macros, insert TEX source code, or associate macros with `Alt-F` keys, etc.

Text insertion. Many macros associated with keys, key combinations, or extended key combinations eventually cause text to be inserted in a document. Any time T-EDIT inserts text into a document, it automatically puts the KEDIT editor into *insert mode* and often repositions the cursor in the proper position for the user to continue. This means that text, which was above and to the right of the cursor at the time the new text is inserted, will be moved to the right. Furthermore, until the user turns insert mode off, additional text typed in will cause any characters above and to the right of the cursor to be shoved to the right.

TEX control sequence insertion macros. The macro associated with the key combination `Ctrl-I` is a redirection macro. It calls upon the selection macro `Ctrl-I.kex`. Although many options are available after `Ctrl-I` is pressed, only one will be discussed in this paper. In particular, suppose that @ (that is, `Shift-2`) is pressed next. A menu of over 1250 Plain TEX and $\mathcal{AMS}$-TEX control sequences and corresponding text insertion macros is initiated. The screen is split into 2 windows. The source program which is being edited appears in the lower window with the cursor under the character it was positioned under when `Ctrl-I` was initially pressed. The user is in what may now be referred to as *menu mode*.

The user sees the first 9 control sequences of the over 1250 which are available displayed in the upper screen window with the first of these control sequences highlighted in red. Also displayed near the top of that window is a message which lists, in a somewhat cryptic form, eighteen options which are available. One of these options, Name, is included to suggest that the user *begin* to type in the name of the control sequence. Suppose that the control word `\begingroup` is needed. After typing `B` and `E`, the user sees

```
\begingroup  \endgroup
```

highlighted in red. In addition, the characters which have been typed in so far, namely `be`, are displayed in the upper portion of the top screen window in order to keep track of what has been typed in. Note that these are lower case letters. In order to have obtained the corresponding upper case letters, the `Shift` key would have had to have been employed. The cursor is still resting beneath the same character as when `Ctrl-I` was pressed. If `Enter` is pressed at this point, then the text

```
\begingroup  \endgroup
```

will be inserted beginning at the current position of the cursor, and the cursor will be repositioned to rest under the second `\`. Any characters on the current line which were above and to the right of the cursor are moved to the right. The menu is still displayed in the top screen window, but there is a new message that informs the user that he or she is in *program mode*. This means that editing can proceed as usual, even though it is actually being accomplished under the control of the macro `Ctrl-I.kex`. The message also instructs the user that to enter menu mode again, press `Ctrl-Enter`, or press `Alt-Esc` to get rid of the menu and return the screen to full screen edit mode.

Now, let's assume that once a desired entry is highlighted, then instead of pressing `Enter`, the user presses `@` (that is, `Shift-2`). Then the highlighted text insertion macro is assigned to some undefined `Alt-F` key. In the bottom window of the screen, a new message appears which tells what `Alt-F` key the macro was assigned to and describes what the macro does. In the case of the preceding example, the displayed message would be

```
Alt-F3 : 'text \begingroup \endgroup'
```

if the text insertion macro was automatically assigned to the undefined `Alt-F` key `Alt-F3`. The user can prohibit the display of such information, then display it again, etc. This time, T-EDIT stays in menu mode. After all, there are probably more text insertion macros which should be assigned to `Alt-F` keys. To escape this mode, `Esc` may be pressed to get back into program mode. In fact, `Esc` may be used at any time the user is in menu mode to return to program mode.

Additional keys which may be used in the menu mode to make a selection are the `Up Arrow` and `Down Arrow` keys, `PgUp` and `PgDn` keys, and the `Ctrl-PgUp` and `Ctrl-PgDn` key combinations. Pressing the `Backspace` key will undo the last key entry, and

pressing the key combination `Alt-Enter` will cause the highlighted macro or text insertion macro to be executed with an automatic cancellation of the menu.

It should be clear that macros which control menus are not selection macros as described earlier. All but 21 of the over 1250 T-EDIT KEXX text macros which can be accessed using the menu just described are generated on the fly. That is, they do not exist at the time they are called upon, but are created using a special code which is present in the menu line containing the desired control sequence, but which is *not* seen by the user. Specifically, a code appears in the first 4 columns of each menu line which instructs T-EDIT how to handle the creation of the text insertion macro, or if the macro is saved on the hard disk instead, then the code `m` is found in column 1 and the name of the macro, which is visible to the user, is found starting in column 69 of the line. Although the other codes available are quite simple, it is far beyond the scope of this paper to describe them all.

Private macros. The key combination `Ctrl-P` will activate a private menu of user defined macros. This menu in used in the same way as described for the menu of TEX insertion macros. However, many of the macros which may be called upon in this menu are much more powerful than any control sequence TEX insertion macros. In addition, the menu may call upon sub-menus, and a couple of extra options may be used in searching for specific macros. Also, in most cases, a fairly lengthy description of what the macro will do is included, and this will be displayed at the bottom of the screen if the user elects to assign the macro to an `Alt-F` key. Furthermore, when called upon, most of these macros lead the user through all of the additional steps necessary to fill in needed data. Some of these macros are powerful TEX code-generating macros which in many cases make use of specially designed TEX macros.

Examples of private macros. Three examples have been included in the Appendix to demonstrate the capabilities of T-EDIT. In each case, a typesetting problem is presented, the steps necessary to generate the required TEX source code using T-EDIT are discussed, and the output obtained from the code is displayed. Please look carefully at these examples so that you can judge for yourself!

At the present time, the private menu refers to a great number of macros which perform remarkable typesetting feats with a minimum amount of effort. In the future, many new private macros will be added to T-EDIT. As a matter of fact, the list will probably continue to grow indefinitely as more and more applications arise. At the same time, many difficult typesetting problems will be reduced to trivial tasks.

Alt-F key management. If the key combination `Alt-E` is pressed, then a number of options are made available. It is now possible to edit or delete `Alt-F` key macro definitions and descriptions, or even enter new ones directly from the keyboard. *Collections* of `Alt-F` key definitions and descriptions may be automatically saved and added to a menu. When the user saves such a collection of `Alt-F` key definitions, he or she is asked to enter a name for the file in which to store the key information as well as a description of the collection. This is then added to a menu so that the entire collection of key definitions and descriptions may be easily reinstated later using that menu. In addition, files containing collections of `Alt-F` key definitions as well as the menu referring to collections of `Alt-F` key files may be edited. The macros associated with `Alt-E` make each of these tasks fairly easy. Because of the capabilities just mentioned, the key assignments which T-EDIT makes at the beginning of each editing session should be adequate for most users.

Letter and mail merge menu. By pressing `Alt-0`, a menu of letter-writing and mail-merge options appears on the screen. The user is aided in writing documents with a minimum amount of effort. Descriptive information concerning a document is recorded in a menu. Later, if the document needs to be located, the desired menu is called upon. Once the information describing the desired file is seen highlighted in red, pressing `Enter` will cause the file to be loaded into the KEDIT editor to be reviewed or revised. Special data-writing macros for mail-merge documents are included. Again, descriptive information concerning specific data sets is inserted in a menu so that the data set may be located easily in the future. Other options allow letter or data menus to be edited. Of course, all documents and data sets are written for use with TEX.

The key combination `Alt-0`, which may seem somewhat out of place, was chosen so that no useful key combination would be wasted. It was included as an additional way of accessing a macro on the hard disk which was actually designed to be initiated from the DOS command line using the Function key `F11`.

Special text insertion keys. Some examples of simple macros associated with keys or combinations of keys are as follows. Keep in mind that T-EDIT is currently used most frequently to enter text which contains mathematical text written for use with Plain TEX or $\mathcal{AMS}$-TEX. It is useful to have additional methods of inserting often-used control sequences so that these control sequences can be entered as easily as possible. When `F1` is pressed, `$$` is inserted in the file beginning at the position of the cursor at the time that the key is pressed. The cursor is positioned under the second `$` sign. The simple macro associated with the Function key `F1` is

```
'insert on';'text $$';'cur left'.
```

Definitions of other simple macros are similar. If the user proceeded to type `E=mc^2`, then after typing the `2`, `$E=mc^2$` would be seen with the cursor positioned under the last `$` sign. Remember, KEDIT was put automatically into insert mode when the `F1` key was pressed. Similarly, pressing `Ctrl-[` causes `{}` to be inserted as text with the cursor repositioned below `}`. The symbol `{` appears on the key referenced by the symbol `[`, so the choice of `Ctrl-[` to represent `{}` is a fairly natural one. Pressing `Ctrl-\` causes `\{ \}` to be inserted into the source file with the cursor resting under the blank space to the right of `{`. Here, since the first character generated is the character `\`, the key combination `Ctrl-\` was chosen to generate this character string.

The macro associated with the key combination `Ctrl-D` is a redirection macro which calls upon the selection macro `Ctrl-D.kex`. When `Ctrl-D` is pressed, a message appears on the screen which instructs the user to press `Esc` to get out of the menu, press `1` for `$$`, press `2` for `$$$$`, or press `3` for `$$ $$` spread out over three lines. Only one of the three options will be discussed. Suppose the user presses 3. Then the following will occur. If the cursor is *not* resting on a blank line, then a new line is first added after the line the cursor was positioned on. Three lines of text are created. The first line contains `$$` left justified, the second line is a blank line, and the third line is `$$`. The cursor is repositioned to appear at the first column of the blank line separating the two `$$` groups. This format is used when mathematical expressions are to be entered in display math style.

The macros associated with the key combinations `Ctrl-G` and `Ctrl-O`, which are discussed next, are redirection macros, but the corresponding macros on the hard disk which they call upon are *not* selection macros or menu-type macros. Details cannot be included in paper as short as the present one.

Greek letters. To get Greek letters quickly, the combination of keys `Ctrl-G` may be pressed followed by one, two, or three of the letters in the control word representing that letter. For example, the text `\varepsilon` is automatically inserted after `Ctrl-T` is pressed followed by `VE`, while the text `\gamma` is automatically inserted after pressing `Ctrl-G` followed by `G`. The instant that the user has typed in enough information to distinguish the desired Greek letter from all others, then the text for that Greek letter is automatically inserted. The user never presses the `Enter` key.

Math delimiters. Opening and closing math delimiters can be obtained by pressing `Ctrl-O`. Suppose that the text

```
\left\lceil  \right\rceil
```

needs to be inserted in a document to create delimiters for a mathematical expression. Assume that the key combination `Ctrl-O` is pressed. A message appears at the top of the screen which instructs the user to press the key which represents the first symbol in the opening delimiter. So suppose that `\` is pressed. At this point, the screen will be split into two windows, with a menu displayed in the upper window. The source file appears in the lower window with the cursor positioned as it was before `Ctrl-O` was pressed. Suppose `LE` is typed next. The code `\left \right` appears highlighted in red in the menu. Assume that `Enter` is pressed. The text `\left` is inserted. However, note that `\right` has not been inserted yet. It has been saved to be inserted later. The menu is still in effect, and a message instructs the user to type the first symbol of the code representing the opening delimiter. That symbol would be `\` for the opening delimiter `\lceil`. So suppose that `\` is pressed next. A matching math opening and closing delimiter code is highlighted in red, but it is not the pair which is currently being sought. However, after pressing `C` and `E`, the desired matching opening and closing delimiter pair is seen highlighted in red. If the `Enter` key is pressed next, then the menu disappears, and it is observed that the TEX code

```
\left\lceil  \right\rceil
```

has been inserted in the source file with the cursor positioned under the first space to the left of the control word `\right`. Note how the control word `\right`, which was not present before, has now been

inserted in the proper position. Incidentally, the requirement that \ be entered as the first symbol in the control word representing the opening delimiter is due to the fact that simple math opening and closing delimiter pairs like () may be used also.

TEXing, Previewing, and Debugging

When a source file is ready to be TEXed, the combination of keys Ctrl-TE is pressed. A check is made to make sure that a \bye or \end statement is included at the end of the program. If there is no such statement, then one is added. The TEX source file is saved, and then TEXed. If no error occurs when the source file is TEXed, then the output is automatically previewed. After the preview is completed, there is a return to the editor and the source file. The cursor is positioned wherever it was when Ctrl-TE was pressed. If an error occurs while TEXing the file, and if the user enters the TEX option e, then the line in the source file where the error occurred becomes the current line in edit mode, and the cursor is positioned at the beginning of this line. Furthermore, the screen is split into two windows, and as much of the *pertinent* log file information as possible is shown in the upper window. If it isn't possible to display all of this information, then the user is advised and informed to press Ctrl-U if there is need to review more of the error message than is shown. If Ctrl-U is pressed, then although the cursor never moves from its current position in the source file, the log file may be reviewed using the Up Arrow, Down Arrow, PgUp, PgDn keys, etc. The key combination Ctrl-TT may be used to TEX a file without previewing the output.

Pressing the key combination Ctrl-TB allows a user to TEX a KEDIT so-called "marked block" in a file. This could include as much or as little TEX source code as desired. In fact, a single character could be TEXed this way. What actually happens is that the block is saved as a TEX source file called \exper.tex with a \bye statement inserted as the last line. This file is then TEXed and previewed. If there is an error, then the offending line in the original file, not the \exper.tex file, becomes the current line in the editor with the cursor positioned at the beginning of that line. The screen is split into two windows, with the error message or a portion of it displayed as was described above.

Pressing the key combination Ctrl-TO does the very same thing as Ctrl-TB with one important exception. Any lines in the file with % in column 80 are also included in the \exper.tex file. Such lines can be marked immediately by pressing Ctrl-M while the cursor is positioned on the line. Later the symbol % may be erased by pressing Ctrl-E while the cursor is positioned on the line. However, in most cases, only certain lines need to be marked in this way, and they will not have to be altered again later.

Pressing the key combination Ctrl-TD will cause all lines in the current source file, from the first blank line at or above the cursor to the bottom of the file, together with any lines marked with a % in column 80, to be TEXed, and previewed, etc.

Pressing Ctrl-C will cause a forward search from the beginning of the file for matching pairs of {}, followed by a backward search from the end of the file for such matching pairs. Pressing the key combination Ctrl-TC followed by F will cause a forward search from the current cursor position to the end of the file for matching pairs of this type, while pressing Ctrl-TC followed by B will cause a backward search from the current cursor position to the top of the file for such pairs. In all cases, the search will end at the first { or } for which there is no match.

If Ctrl-S is pressed, then the MicroSpell spelling checker is activated. Up to about 90% of the spelling errors in the document are usually spotted and corrected this way. The key combination Ctrl-TP will cause the output to be printed, and Ctrl-TH will cause the dvi file associated with the source file to be copied to a diskette.

In case a user cannot recall all of the pre-defined key definitions, Alt-H brings up a menu of all such key assignments the user may not be familiar with. Once the given key and description are shown highlighted in red, the key can be accessed by simply pressing the Enter key! By pressing Ctrl-H, a menu comprised of a subset of the key definitions just mentioned, which are TEX related, is initiated.

Future of T-EDIT

T-EDIT is still in its infancy. It will continue to grow and become more sophisticated in the future regardless of whatever else happens. Hopefully, other individuals will become interested in T-EDIT, and perhaps a method for the distribution of T-EDIT software will become available.

Bibliography

Riley, Don L. and Brad L. Halverson, "Creating an Efficient and Workable PC Interface for TEX", *TUGboat*, **10** (4), pages 751–759, 1989.

Appendix

Example 1

Suppose that a user wants to display the long division of the polynomial

$6x^6 - 15x^5 - 34x^4 + 36x^3 - 14x^2 - 7x + 7$

by the polynomial

$2x^2 + 3x - 2.$

The user presses `Ctrl-P` to get into the private macro menu. After `POL` has been typed in, the user sees the entry which says `polynomial division` highlighted in red. If the user presses `Enter`, then the first thing which is done by T-EDIT is to search the TEX source file for an `\input` statement for file `\polydiv.mac`, which contains the TEX macros which will be needed. If no such `\input` file is found, then one will be added automatically to the source file at the beginning of the file or following the last `\input` statement found, if any. While this is accomplished, the cursor appears to stay in precisely the same position that it was in before `Enter` was pressed. Next, the name of the variable to be used is requested. This could be a Greek letter, etc. if desired. Suppose that x is entered. The user is then asked to insert the coefficients of the denominator. Here, that would mean that `2 3 -2` is typed in and then entered. Now the coefficients of the numerator are requested. For the current problem, the user would type `6 -15 -34 36 -14 -7 7` and enter it. T-EDIT causes all of the additional numerical information required to be computed. Line after line of TEX code is automatically generated until all necessary code has been inserted. The code generated by T-EDIT is shown below.

```
\polydiv
\lp 3x^{4}\m 12x^{3}\p 4x^{2}\p 0x\m 3
\hbar{15}
2x^{2}+3x-2\vbar\lp 6x^{6}\m 15x^{5}\m 34x^{4}\p 36x^{3}\m 14x^{2}\m 7x\p 7\crn{3}
\lp 6x^{6}\p 9x^{5}\m 6x^{4}\ubar{4}{5}{5}
\lm 24x^{5}\m 28x^{4}\p 36x^{3}\crn{5}
\lm 24x^{5}\m 36x^{4}\p 24x^{3}\ubar{5}{6}{7}
\lp 8x^{4}\p 12x^{3}\m 14x^{2}\crn{7}
\lp 8x^{4}\p 12x^{3}\m 8x^{2}\ubar{8}{5}{11}
\lm 6x^{2}\m 7x\p 7\crn{11}
\lm 6x^{2}\m 9x\p 6\ubar{11}{6}{13}
\lp 2x\p 1
\endpolydiv
```

The output which will be produced by the code is shown below.

$$
\begin{array}{rl}
 & 3x^4 - 12x^3 + \ 4x^2 + \ \ 0x - \ \ \ 3 \\
\cline{2-2}
2x^2 + 3x - 2 \,\big| & 6x^6 - 15x^5 - 34x^4 + 36x^3 - 14x^2 - 7x + 7 \\
 & \underline{6x^6 + \ 9x^5 - \ 6x^4} \\
 & \quad - 24x^5 - 28x^4 + 36x^3 \\
 & \quad \underline{- 24x^5 - 36x^4 + 24x^3} \\
 & \qquad\qquad 8x^4 + 12x^3 - 14x^2 \\
 & \qquad\qquad \underline{8x^4 + 12x^3 - \ 8x^2} \\
 & \qquad\qquad\qquad - \ 6x^2 - 7x + 7 \\
 & \qquad\qquad\qquad \underline{- \ 6x^2 - 9x + 6} \\
 & \qquad\qquad\qquad\qquad 2x + 1
\end{array}
$$

The user can position the output on the page however desired. In the case of the example above, the display has simply been indented by the default indentation amount.

Example 2

A user desires to create a short table of Laplace transforms of functions. Assuming that the appropriate T-EDIT macro has not already been assigned to an `Alt-F` key, the first step in creating the table would be to press `Ctrl-P` to access the private macro menu. After `TA` has been typed, the desired entry is highlighted in red. It is listed in the form `table, math` together with some additional information which identifies the table style. Beneath this entry, other styles of math tables are listed.

To initiate the macro, the user presses `Enter` (or `Alt-Enter` if the menu is to be exited after the selection of the macro). A check is made to see if the user has included an `\input` file for the required TEX macro `\mathtabl.mac`. If no such statement is found, then one is added after the last `\input` statement in the file, or at the beginning of the file if there is no `\input` statement. A beep sounds and a message appears which requests the user to enter the title on one line, separating lines of the title with `\cr`, or to press `Enter` to quit. Suppose that `Laplace Transforms\cr of Functions` is typed in and entered. A new line is added if the line upon which the cursor is resting is not a blank line. The text below is inserted, a beep sounds, and a message appears requesting the user to either enter the amount to spread the table or to press `Enter` to accept the default of 100pt.

```
\mathtable{Laplace Transforms\cr of Functions}
```

Assume that `Enter` is pressed to accept the default. The symbols `{}` are added to the text shown above. There are three more input parameter values which the user may enter or accept default values for. The first of these parameter values is the amount of indentation from the left margin, and has a default value of 1truein associated with it. The other two parameters represent strut heights and depths which will be used in the construction of the main part of the table. Suppose that the default value is accepted for each of these parameters. After the last time `Enter` is pressed to accept a default value, a new line is automatically added, the symbol `|` is inserted, and the cursor is positioned two spaces to the right of it. So far,

```
\mathtable{Laplace Transforms\cr of Functions}{}{}{}{}
  |
```

has been generated. Now, the user is asked to enter a heading for the first column and press the Tab key when the entry is completed. Suppose that `f(t)` is typed, and then the Tab key is pressed. The following text is displayed with the cursor positioned two spaces to the right of the last `|` shown.

```
\mathtable{Laplace Transforms\cr of Functions}{}{}{}{}
  | f(t)                          |
```

Next, a message requesting the entry of the column heading for the second column appears. The message also includes instructions to press the Tab key when the entry is completed. Assume that `{\cal L}(f)` is typed in and the Tab key is pressed. The text which has been inserted after the Tab key was pressed is shown below.

```
\mathtable{Laplace Transforms\cr of Functions}{}{}{}{}
  | f(t)                          | {\cal L}(f)                      |
  |
```

The symbol `|` has been automatically inserted twice more, and the cursor is positioned on the last line shown, two spaces to the right of it. The user is requested to insert the entry for the first column and press the Tab key when that entry is completed, or to press @ to finish the creation of the table. Suppose that 1 is typed in and the Tab key is pressed. So far, the following text has been added to the TEX source file .

```
\mathtable{Laplace Transforms\cr of Functions}{}{}{}{}
  | f(t)                          | {\cal L}(f)                      |
  | 1                             |
```

A space separates the last symbol `|` on the right from the current position of the cursor.

The text `1\over s` is typed in next, and the Tab key is pressed. The text below is present after the Tab key is pressed, with the cursor positioned one space to the right of the symbol `|` on the last line.

```
\mathtable{Laplace Transforms\cr of Functions}{}{}{}{}
  | f(t)                          | {\cal L}(f)                    |
  | 1                             | 1\over s                       |
  |
```

Now that the idea is clear, let's look at the keystrokes needed to complete the table. `Tab` will be used to represent the `Tab` key.

```
e^{at} Tab 1\over{s-a} Tab \sin at Tab a\over{s^2+a^2} Tab \cos at
Tab s\over{s^2+a^2} Tab \sinh at Tab a\over{s^2-a^2} Tab \cosh at Tab
s\over{s^2-a^2} Tab @
```

All of the necessary code to generate the Table has been entered and appears below.

```
\mathtable{Laplace Transforms\cr of Functions}{}{}{}{}
  | f(t)                          | {\cal L}(f)                    |
  | 1                             | 1\over s                       |
  | e^{at}                        | 1\over {s-a}                   |
  | \sin at                       | a\over {s^2+a^2}               |
  | \cos at                       | s\over {s^2+a^2}               |
  | \sinh at                      | a\over {s^2-a^2}               |
  | \cosh at                      | s\over {s^2-a^2}               |
\endmathtable
```

Note that the required control word `\endmathtable` was automatically inserted on the last line. In addition, one extra line has been added following the last line shown, and the cursor is positioned under the first column of that line. The table produced by the code is illustrated below.

Laplace Transforms of Functions

$f(t)$	$\mathcal{L}(f)$
1	$\dfrac{1}{s}$
e^{at}	$\dfrac{1}{s-a}$
$\sin at$	$\dfrac{a}{s^2+a^2}$
$\cos at$	$\dfrac{s}{s^2+a^2}$
$\sinh at$	$\dfrac{a}{s^2-a^2}$
$\cosh at$	$\dfrac{s}{s^2-a^2}$

Example 3

Consider the problem of displaying a linear system of equations. In this example, it will be assumed that the T-EDIT macro has already been assigned to an `Alt-F` key, say `Alt-F5`, using the private menu. Then towards the bottom of the screen, the message

```
*** Use Ctrl-V to toggle info. display. Use Alt-5 to clear keys and screen ***
```

appears together with information for `Alt-F` keys which have been defined. For `Alt-5`, the user would see the following line displayed.

```
Alt-F5 :  systems of equations
```

Assume the key combination `Alt-F5` is pressed. As in the case of the previous examples, the `\input` statement for the macro `\sysequa` is inserted in the source code if it does not already appear. A beep is heard, and a message appears towards the top of the current screen window which instructs the user to enter each variable name to be used *or* enter the name of the subscripted variable name which will be employed followed by the number of unknowns. Pressing `Enter` is mentioned as a method of terminating execution. For this example, suppose that `\alpha 4` is typed in and entered. If the cursor is resting on a line which is not completely blank, then a new line is added. The code

```
\sysequa
```

appears left justified as shown on that line. In addition, a beep is heard and a message appears towards the top of the current screen window which instructs the user to enter four coefficients, followed by the entry to appear on the right-hand side of the equals sign for this row equation. Entering too little information or too much information for this row equation will cause several beeps to sound, together with a message instructing the user to enter values for this row equation again. If `1 35 -3 2 6` is entered then the following code will be seen.

```
\sysequa
  \lp \alpha_{1} \p 35\alpha_{2} \m 3\alpha_{3} \p 2\alpha_{4} \eq{6}
```

Of course, a beep is heard and the same message appears as before. Continuing, let's assume that `15 -4 21 9 5` is entered, followed by `-2 0 2 4 8`, and then `1 3 0 -1 10`. Finally, `f` is entered to finish. The code which has been generated by T-Edit is shown below.

```
\sysequa
  \lp \alpha_{1} \p 35\alpha_{2} \m 3\alpha_{3} \p 2\alpha_{4} \eq{6}
  \lp 15\alpha_{1} \m 4\alpha_{2} \p 21\alpha_{3} \p 9\alpha_{4} \eq{5}
  \lm 2\alpha_{1} \zero \p 2\alpha_{3} \p 4\alpha_{4} \eq{8}
  \lp \alpha_{1} \p 3\alpha_{2} \zero \m \alpha_{4} \eq{10}
\endsysequa
```

Note that `\endsysequa` has been automatically added to the code. Furthermore, an additional line has been added following this line, and the cursor appears at the first column of the new line.

The system of equations generated may be displayed wherever desired on the page. If `$$` had been typed in before the macro was initiated, and if `$$` had been entered on the line following the command word `\endsysequa`, then the output obtained would be the same as that displayed below.

$$\begin{aligned}
\alpha_1 + 35\alpha_2 - \ 3\alpha_3 + 2\alpha_4 &= \ 6 \\
15\alpha_1 - \ 4\alpha_2 + 21\alpha_3 + 9\alpha_4 &= \ 5 \\
-\ 2\alpha_1 \qquad + \ 2\alpha_3 + 4\alpha_4 &= \ 8 \\
\alpha_1 + \ 3\alpha_2 \qquad - \ \alpha_4 &= 10
\end{aligned}$$