In practice most users start with WordPerfect and sometimes end with TEX.

Intermigration tools between these systems are therefore useful. K-Talk is a program that translates WordPerfect files into TEX files. This report aims to provide an answer to the question 'Is K-Talk a good tool?' Apart from 'the answer' — if any black and white answer is possible — we constructed a test collection of judiciously chosen document elements such that comparison can be made easily with future releases of K-Talk or similar products.

Furthermore is must be noted that translation of a WordPerfect document of only a few pages will produce a TEX file, preceded by a dozen or so pages of macros; moreover a macro library of considerable size is used. The average TEX user will usually not understand those macros. Adaptation of the produced document is not always simple. We also note that processing of the translated `.tex` file by e.g. LATEX or $\mathcal{AMS}$-TEX is by no means trivial.

Training

Making Paragraphs

Alan Wittbecker

TEX, as perhaps you already know, is a typesetting program for the production of beautiful pages. Using TEX to produce beautiful pages is easy if you let TEX make the design decisions already built into the program. TEX is easier to understand, however, if you have an appreciation of the history of typography and book production, not to mention computer programming.

This article, as the first of a series, presents TEX in the simplest way — as a set of instructions for typesetting text. (The tutorial on page 276, by contrast, concentrates on TEX as a computer program.) TEX instructions are added to a file, called the source file, containing text. The text has been entered with words separated by spaces and groups of words separated by punctuation and blank lines, which represent the boundaries of phrases and paragraphs.

TEX instructions describe the procedure that TEX follows when it formats the text. The instructions and the text are entered using keyboard characters (ASCII characters), so that the file can be transferred easily to other computers. TEX formats a source file to a device-independent file (DVI file) that can be printed (after going through a DVI translator and output driver) on dot-matrix printers, laser printers, or typesetters.

A TEX instruction requires a special character, the backslash (\), to be recognized as an instruction. The body of the instruction is composed of alphabetic characters, usually a word; each instruction, in general, is ended with a space. Instructions are simply entered in the text.

Instructions are gobbled up — it is permissible to anthropomorphize TEX — when the text is formatted. Characters are read and printed as characters, unless they are special, like the backslash or percent sign (%), which is the comment character; but, even these can be printed with a specific instruction. Extra blanks between words are discarded because TEX calculates an optimum interword space — therefore, you can use blanks to arrange the source file. The carriage return from the terminal is converted to a blank.

TEX Makes Paragraphs

TEX considers the paragraph the basic unit of production. A paragraph is a short composition consisting of a group of sentences. The clear separation of paragraphs can distinguish thoughts, clarify content, and increase comprehension. A paragraph is commonly indicated by starting a new line and indenting that line. Sometimes space between paragraphs is also used to distinguish them.

After TEX reads text into its memory — by the mouthful! — paragraphs are examined for goodness according to a mathematical standard of beauty (based actually on calculations of "badness"), and then separated into lines.

TEX recognizes the beginning of a paragraph by reading an alphabetic or numeric character. The paragraph instruction itself (`\par`), however, comes at the end of a paragraph, instead of the beginning. All other TEX instructions precede the text they describe. A blank line also causes the end of a paragraph; actually, two consecutive keyboard carriage returns are translated as a `\par` instruction. Thus, the simplest source file only needs blank lines separating paragraphs. Figure 1 lists a source file. Notice the use of the percent sign to comment out information — nothing that follows a percent sign on a source line is even read.

```
%     figure 1. paragraphs
\TeX\ is part of a long tradition of
```

```
putting printed words on a page.
Many conventions of book and journal
production have roots in Mesopotamia
and Egypt, Greece and Italy.

Egyptian papyrus scrolls were written in
hieroglyphics in vertical columns
separated by thin black lines; sometimes
illustrations accompanied the text along
the top or bottom of the scroll, marked
off from text by double ruled lines.
As illustration assumed more importance,
it was placed within the text.
This format persisted through Greek and
Roman manuscripts to medieval and modern
books.

Although Greek scrolls were also written
in columns, characters were presented
continuously, in capitals, without breaks
between words.
Punctuation was usually nonexistent.
Breaks in thought were sometimes indicated
by an underlining stroke (known as a
paragraphos) or by a small blank space.
```

Figure 1. Paragraphs Source File

The output from this source file produces a formatted series of paragraphs (Figure 2). The lines in the source file are concatenated to fill in the full measure of the text area (6.5 inches). Additional spacing may be added between words to right justify them. Some words may be hyphenated to minimize the amount of spacing between words. The last line is filled in automatically with blank space.

TEX automatically makes basic decisions, called defaults, that determine the margins of a page, the kind and size of type, and the shapes of paragraphs. These decisions use default values. For paragraphs, the default amount of indention is twenty points; the default space between paragraphs is zero points. These defaults can be changed with specific instructions.

Paragraph Indent

The paragraph indention can be controlled with the `\parindent` instruction. The syntax is `\parindent` ⟨*dimen*⟩ where ⟨*dimen*⟩ is a parameter that has two parts: a number and a unit of measure. (The value of that number can be positive or negative — a negative value creates a hanging indention.)

A point is a unit of measure (1 inch = 72.27 points) developed by le Juene in 1737 for the metal type invented by Johann Gutenberg in 1440 — those small rectangular blocks of type required a fine measuring system. Type is traditionally measured in points. TEX performs measurements in points (`pt`), although dimensions can also be specified in `cm` (centimeters), `in` (inches), `mm` (millimeters), `em` (1 em ≈ width of capital letter M), `ex` (1 ex ≈ height of lower case x), as well as other units.

The indention can be suppressed for a paragraph with the `\noindent` instruction before the first text character. Then the first line begins at the left margin. Subsequent paragraphs are typeset with the normal indention. A special instruction for modifying a paragraph, such as `\noindent`, can also act to initiate a paragraph.

Paragraph Skip

The amount of vertical space between paragraphs can be controlled with the `\parskip` instruction. The syntax for this instruction is `\parskip` ⟨*dimen*⟩ where ⟨*dimen*⟩ is any number and valid unit of measure. The default value is `0pt`. When paragraphs are indented, extra space is optional; if the indention is set to zero, then extra space between them may be necessary.

Excerpt Paragraphs

An excerpt is a form of secondary text set off from the main text by margin indentions (and sometimes by type size or spacing before and after the text). The left margin is changed with a `\leftskip` instruction; the right margin with a `\rightskip` instruction. The default for both is zero. The syntax for each is `\leftskip` ⟨*dimen*⟩ where ⟨*dimen*⟩ can be any number and valid unit of measure.

A `\narrower` instruction indents the entire paragraph at *both* the left and right margins by the value of the `\parindent`. These instructions change the shape of all paragraphs that follow; that is, the values are *not* reset automatically. The effect of an unlimited instruction can be limited by putting that instruction within special delimiters. The open and close curly braces ({}) are TEX delimiters. A close brace must always match an open brace.

When these changes are included in the source file, the formatted version becomes more complex (see Figure 3). Notice the difference between the plain paragraphs in Figure 2 and the modified paragraph in Figure 4.

```
% Figure 3.  modified paragraphs
\parindent 0pt
\parskip 6pt
```

```
Papyrus, prepared from sliced reeds
pressed and glued together, was the
most commonly used book material in
Greece and Rome.
A crude vellum, made from animal
skins, had been known in old Egypt.

Pliny, the historian, relates that
Eumenes, King of Pergamum, wanted a
fine library for his city, but King
Ptolemy of Egypt, to avoid any
rivalry to the great library of
Alexandria, forbade the export of
papyrus to Pergamum (circa 170 B.C.).
Not to be foiled, Eumenes sponsored
the development of a finer, two-sided
vellum as a writing surface.

{\leftskip 10pt\rightskip 10pt
The library at Pergamum later became
an important center of culture.
It had 200,000 volumes when Antony
presented it as a gift to
Cleopatra---who made it part of the
Alexandrian library.\par} %par must
% be in braces for indents to work!

Moveable type has an honorable lineage.
A clay disk, dating from 1500 B.C.,
was found in the ruins of the palace
of Phaistos on Crete.
Later, in China in A.D.~1041, Pi-Sheng
developed type characters from hardened
clay.

Clay, however, did not hold up well
under repeated impressions.
By 1397 in Korea, type characters were
being cast in bronze.
Then, in 1440, Johann Gutenberg
demonstrated the commercial possibilities
of graphic reproduction with metal type.
\par
```

Figure 3. Altered Paragraphs Source

Hanging Paragraphs

Hanging paragraphs are the inverse of normal ones, where the first line is indented but the following ones are a full-measure wide. In a hanging paragraph, the first line is full-measure and the run-over lines are indented, usually by the amount of space of a paragraph indent. You must specify the indention value.

Hanging paragraphs are created by typing `\hangindent` ⟨*dimen*⟩ at the beginning of an nonindented paragraph (where `\parindent 0pt` or `\noindent` is used). For example, if ⟨*dimen*⟩ is given as `20pt`, all but the first line of the paragraph will be indented twenty points from the left margin; the first line will start at the left margin. To make a series of hanging paragraphs, you must end the previous paragraph, then state the hangindent, and finally start the paragraph with a `\noindent` (unless `\parindent 0pt` is set).

The number of full-measure lines is determined by `\hangafter 1`, the default. The parameter number determines the number of lines left full-measure wide. The number can be made negative with a minus sign — in fact, a `-1` and a `\hangindent 20pt` gives a normal paragraph). A sample source file is shown in Figure 5 and formatted in Figure 6. Note how instructions can be doubled.

```
% Figure 5. Hanging Paragraphs
\hangindent 30pt\noindent
Alphabet length. The horizontal measure,
in points, of the lower case alphabet
set in type of one size and face
(sometimes used to describe an optimum
width, e.g., 1.5 alphabet lengths).

\hangindent 30pt\noindent
Alignment. The way text lines up on a
column: align left (or flush left or
raggedright), align center, align right,
or justify (flush right and left).

\hangindent 30pt\noindent
Ascender. The part of a lowercase letter,
such as b or d, that extends above the
x-height (the height of a letter x).

\hangindent 30pt\noindent
Base line. An imaginary horizontal line
connecting the bottoms of capital letters
(not inclusive of the descenders of lower
case letters).

\hangindent 30pt\noindent
Body type. Type used for the text of a
work, as distinguished from display type,
which is used for chapter headings or
titles. The optimum size ranges from 10
to 12 points depending on style and use.
\par
```

Figure 5. Hanging Paragraphs Source

TEX is part of a long tradition of putting printed words on a page. Many conventions of book and journal production have roots in Mesopotamia and Egypt, Greece and Italy.

Egyptian papyrus scrolls were written in hieroglyphics in vertical columns separated by thin black lines; sometimes illustrations accompanied the text along the top or bottom of the scroll, marked off from text by double ruled lines. As illustration assumed more importance, it was placed within the text. This format persisted through Greek and Roman manuscripts to medieval and modern books.

Although Greek scrolls were also written in columns, characters were presented continuously, in capitals, without breaks between words. Punctuation was usually nonexistent. Breaks in thought were sometimes indicated by an underlining stroke (known as a paragraphos) or by a small blank space.

Figure 2. Formatted Paragraphs from Figure 1.

Papyrus, prepared from sliced reeds pressed and glued together, was the most commonly used book material in Greece and Rome. A crude vellum, made from animal skins, had been known in old Egypt.

Pliny, the historian, relates that Eumenes, King of Pergamum, wanted a fine library for his city, but King Ptolemy of Egypt, to avoid any rivalry to the great library of Alexandria, forbade the export of papyrus to Pergamum (circa 170 B.C.). Not to be foiled, Eumenes sponsored the development of a finer, two-sided vellum as a writing surface.

The library at Pergamum later became an important center of culture. It had 200,000 volumes when Antony presented it as a gift to Cleopatra—who made it part of the Alexandrian library.

Moveable type has an honorable lineage. A clay disk, dating from 1500 B.C., was found in the ruins of the palace of Phaistos on Crete. Later, in China in A.D. 1041, Pi-Sheng developed type characters from hardened clay.

Clay, however, did not hold up well under repeated impressions. By 1397 in Korea, type characters were being cast in bronze. Then, in 1440, Johann Gutenberg demonstrated the commercial possibilities of graphic reproduction with metal type.

Figure 4. Altered Paragraphs from Figure 3.

Alphabet length. The horizontal measure, in points, of the lower case alphabet set in type of one size and face (sometimes used to describe an optimum width, e.g., 1.5 alphabet lengths).

Alignment. The way text lines up on a column: align left (or flush left or raggedright), align center, align right, or justify (flush right and left).

Ascender. The part of a lowercase letter, such as b or d, that extends above the x-height (the height of a letter x).

Base line. An imaginary horizontal line connecting the bottoms of capital letters (not inclusive of the descenders of lower case letters).

Body type. Type used for the text of a work, as distinguished from display type, which is used for chapter headings or titles. The optimum size ranges from 10 to 12 points depending on style and use.

Figure 6. Hanging Paragraphs from Figure 5.

Item Paragraphs

Items are hanging paragraphs that "hang off" an identifier. The syntax for this instruction is `\item{\signif}` where ⟨*signifier*⟩ is any letter, number, or symbol with optional punctuation; the braces must be included if the ⟨*signifier*⟩ is more than one character.

A second level of indention for itemized lists is given by `\itemitem`, which indents twice the `\parindent` value. These instructions automatically end the previous paragraph. Refer to Figure 7 for an example.

```
% Figure 7  Items
\parskip 9pt  % spaces between pars
\item{1.}% curly braces contain number
Skillin, Marjorie, Robert Gay, et al.
1964.
Words Into Type.
New York: Appleton-Century-Crofts.
\item{2.}
Carter, Rob, Ben Day, and Philip Megs.
1985.
Typographic Design: Process and
Communication.
New York: Van Nostrand Reinhold Co.
\par
```

Figure 7. Item Paragraphs Source

Items are useful for lists, outlines, and bibliographies. Figure 8 shows a bibliography.

1. Skillin, Marjorie, Robert Gay, et al. 1964. Words Into Type. New York: Appleton-Century-Crofts.

2. Carter, Rob, Ben Day, and Philip Megs. 1985. Typographic Design: Process and Communication. New York: Van Nostrand Reinhold Co.

Figure 8. Formatted Items

The instructions presented in this article create paragraphs. Therefore, you should remember to end each one with a `\par` instruction or a blank line.

There is a lot more to paragraphs, including ragged margins, repetition of instructions for each paragraph, and special shapes, but that will be presented much later. The next of these training tutorials will address the contents of paragraphs: special characters, accents, fonts, and lines.

Macros

A Tutorial on `\futurelet`

Stephan v. Bechtolsheim

This is the second in a series of tutorials by this author. This time we will deal with `\futurelet`, a rather interesting instruction which causes many people unnecessary difficulties. This article is condensed from a draft of my books *Another Look at TEX*. See the end of this article for more information about the books.

Introduction

The `\futurelet` primitive is a TEX instruction allowing the user to look ahead. The term "look ahead" means that TEX will look at a future token and provide a copy of that token **without** absorbing it, i.e. without removing that token from the main token list. This operation allows the programmer to perform a test for "what token is coming" (to express it in a rather informal way) on the main token list. The token looked at through `\futurelet` will be removed later, typically as part of an argument of a later macro call as we will see shortly. It is **not** removed by the action of the `\futurelet` primitive.

Let us be more precise now; the `\futurelet` instruction has the following format:

`\futurelet` ⟨token$_1$⟩ ⟨token$_2$⟩ ⟨token$_3$⟩

Here is what TEX will do:

1. TEX will execute a `\let` ⟨token$_1$⟩ = ⟨token$_3$⟩. We therefore have generated a copy of ⟨token$_3$⟩ stored under the name of ⟨token$_1$⟩.
2. TEX removes ⟨token$_1$⟩ from the main token list.
3. TEX expands ⟨token$_2$⟩. This token is for all practical purposes a macro with the following properties:
   (a) The macro will use ⟨token$_1$⟩, which is a copy of ⟨token$_3$⟩, to find out what ⟨token$_3$⟩ is, in other words what token is to be expected later.
   (b) It will cause another macro to be expanded which will ultimately absorb ⟨token$_3$⟩. This other macro ordinarily depends on what ⟨token$_1$⟩ is.

There are many applications of `\futurelet`. We will here present only one example, although we will present it in quite some detail so the user will know how to apply `\futurelet` in different circumstances.