February 23, 1994

FROM: Jeffrey McArthur

SUBJECT: LitTEX Beta Test

Description of What LitTEX Does

The input file is converted into a file suitable for typesetting with TEX. All typesetting is currently done with plain TEX.

Installation

If you are reading this you have successfully uudecoded and unzipped the file. All that remains is to copy littex.mac to a directory that is searched when TEX is run. If you are using emTeX this would be the directory specified in the TEXINPUT environment variable. Place littex.exe in a directory in your path.

Manifest

The zip files contains the following files:

lt_doc.tex	7160	2-23-94	2:50a	Documentation on LitTEX, you are reading it.
lt_doc.dvi	8916	2-23-94	2:50a	processed version of lt_doc.tex
littex.exe	17792	2-23-94	1:24a	Program to do conversions
littex.mac	13816	2-23-94	12:51a	Macros needed to typeset output
l.tex	20511	2-23-94	1:17a	output of littex.exe on littex.mac
l.dvi	48944	2-23-94	12:54a	processed version of l.tex

Indentation

Currently tabs are assumed to be set to four spaces. A future version will support allowing the user to change this. Code indentation is maintained.

Comments

One of my major complaints with tgrind, fweb, and noweb was the processing of comments. LitTEX processes comments quite differently from other programs. There are four categories of comments: full line comment, comment to the end of line, comment of the end of line, and formatting specification.

Full line comments begin with a comment character, usually %, in the left most column. Lines of this type are typeset in italic with the leading percent printed. Special characters are converted to codes to allow them to print.

Comments to the end of line begin with a comment character, %, and have text following them. These lines are split into two parts. The code part is typeset normally, but the comment section is typeset in italic flush right.

Comments that occur at the end of line are used to make code more legible. They are not moved to the right hand margin but left following the text.

LitTEX uses special comments to for formatting instructions. Currently there are eight formatting comments:

%*	Start of major section
%@	Start of minor section
%!	continuation line
%#	omitted line, useful for version control statements and such
%|	verbatim line, useful for included text. Typeset in monospace.
%×4	hairline rule
%×16	1 point rule
%×64	double 1 point rules.

Major and Minor Sections

Major sections start with a comment that begins in the left hand column and is followed by an asterics, *. The rest of the line is assumed to be the name of the major section. Sections are numbered automatically starting at one. At least two major sections need to be defined for an index to be generated. Minor section use a @ instead of *. Minor sections start over from one at the start of each major section. It is customary to follow a major or minor section with some explanatory text. This can be any valid TEX statements. Each line in the continuation text must begin with a %! in the left hand column. Below is an example of how a major section would be coded:

```
%* Example.
%! The text following this would be typeset as a paragraph
%! in \TeX.
```

%! can be used just about anywhere to place a paragraph in the typeset documentation.

Major section force a new page. The title of the major section is typeset in bold face on a line by itself. Minor sections are a considered a good place to break a page. Text following the minor section head will run-in. Any rules directly preceding a major or minor section will be suppressed.

Rules

Occasionally it is useful to place a rule. This would be similar to the format used in Appendix E of The TeX Book. LitTEX supports three types of rules: hairline, single one point rule, and double one point rules. Rules are defined as a sequence of percent characters. Sixty-four, 64, or more percent signs that start at the left hand column define a double rule. Sixteen to sixty-three percent signs define a one point rule. Four to fifteen define a hairline rule.

The constants chosen for for the rules are not arbitrary numbers. They show a bias towards an Emacs style editor. Control-U multiply the current repeat count by 4. The default repeat count is one. So to get a hairline rule is a simple Control-U followed by a percent. A Control-U, Control-U, percent generates a thicker rule, and so on. Visually this is also easy to understand in the ASCII input file.

Rules are considered a good break point. If the rule breaks to the top of a new page it will be suppressed. Rules will never occur at the top or the bottom of a page.

Omitted Lines

Some comments should not be reflected in the typeset document. For example the version control keyword statements do not really have a meaning in the final output. Lines can be omitted by placing a %# in the first column.

Verbatim Lines

Occasionally it is useful to include a line of text in a verbatim mode. Lines of this type are typeset in a monospace font. This is accomplished by using a %| in the left most column.

Blank Lines

Blank lines are replaced with a medium skip. Blank lines are suppressed prior to and following rules. They are also suppressed before major and minor sections.

Changes to Catcodes

Currently LitTEX does a limited processing of catcode changes. The only changes that are tracked are ones that start in the left hand column. Also the assignment value must be a decimal number.

Example

LitTEX.mac is the macro file needed to typeset a file that has been processed by the LitTEX program. The macros use almost all the features mentioned above.

Limitations

Currently lines are limited to 255 characters.

Bugs

Currently LitTEX has two known bugs.

If the input file has a length of zero bytes the program goes into an infinite loop and never exits.

If a $ is in a comment it is printed as £.

\␣ used in a macro definition or comment may not typeset properly.

To-Do List

This is the list of planned enhancements:

Keep track of `\def`, `\edef`, `\xdef`, `\gdef`, and `\chardef`. This would allow underlining entries in the index showing where a control sequence is defined.

Catcode changes should support names like `\active`, `\other`, and `\letter`.

Questions

Should pretty-printing include reformatting? Should `\if` sequences be broken down and indented to show the structure? This processing would be quite simple to add to the current program. But is it desirable? Should it be a command line option?