About \TeX\ FPC. \TeX\ FPC is a port of Donald E. Knuth's typesetting program \TeX, version 3.141592653 from February 2021 to Free Pascal (FPC) and Unix. To help you identify the differences of \TeX and \TeX FPC, the numbers of modified modules carry an asterisk. Letters in the left margin indicate the reason for a change. They mean:

- **E** fixes an error in \TeX 82
- **F** adds a feature as suggested by Knuth
- **X** describes an FPC extension
- **U** necessary change in Unix
- **u** enhances usability in a Unix environment

### E
1. \TeX 82 deletes area and extension of an input file name and then only shows the base name of the file during error recovery.
2. \TeX 82 prunes discardable nodes from the beginning of a new line until it reaches a nondiscardable node. This might leave you with an empty box resulting in an `underfull box` warning. Btw, I discovered this bug while trying to prove the line breaking algorithm, not while plain testing it. If you have time, prove, if not test.
3. \TeX fails to respect end of file (Control-D) from terminal input during debug dialog.
4. Igor Liferenko reported an extra space in the transcript file after the user switched to `/batchmode` during error recovery.

### F
1. \TeX FPC treats the command line as the first input line;
2. \TeX FPC starts `ed`, the unix system editor, if the user types `E` during error recovery.
3. You can interrupt \TeX FPC by typing `Control-C`.

### P
1. \TeX 82 assumes that the terminal input file is positioned before the first character after being opened, whereas \TeX FPC assumes that it is positioned at the first character, thus complying with the Pascal standard.
2. The names of the standard text files must occur in the program header whenever they are used.
3. The standard text files must not be declared. Declared files with the name of the standard text files are new internal files.
4. The program must not open the standard text files.

### X
1. FPC's extensions are needed to specify a file name at run time, to check the existence of files and to access the system date and time. Identifiers from FPC Pascal are prefixed with `fp` to help distinguish them from Pascal and `WEB` identifiers and to avoid name clashes. Furthermore all FPC Pascal identifiers will appear together in the index.

### U
1. The Unix file separator is `/` instead of `:`.

### u
1. On exit, \TeX FPC passes its 'history' to the operating system. This integer is zero when everything is fine, one when something less serious like an overfull box was detected, two when an error happened like an undefined control sequence, and three when the program aborted because one of its tables overflowed or because it couldn't find an input file while running in batch mode.
2. Valid input characters are the 94 visible ASCII characters together with the three control characters horizontal tabulator, form feed, and space.
3. Terminate last line on terminal. This is Unix, not DOS!
4. Teach \TeX and user how to end the terminal input by `Control-D`. 

February 2021
Wolfgang Helbig, Programmierer
Waiblingen, Baden-Württemberg
helbig@mailbox.org

2* The present implementation has a long ancestry, beginning in the summer of 1977, when Michael F. Plass and Frank M. Liang designed and coded a prototype based on some specifications that the author had made in May of that year. This original prototype \TeX included macro definitions and elementary manipulations on boxes and glue, but it did not have line-breaking, page-breaking, mathematical formulas, alignment routines, error recovery, or the present semantic nest; furthermore, it used character lists instead of token lists, so that a control sequence like \texttt{\textbackslash halign} was represented by a list of seven characters. A complete version of \TeX was designed and coded by the author in late 1977 and early 1978; that program, like its prototype, was written in the SAIL language, for which an excellent debugging system was available. Preliminary plans to convert the SAIL code into a form somewhat like the present “web” were developed by Luis Trabb Pardo and the author at the beginning of 1979, and a complete implementation was created by Ignacio A. Zabala in 1979 and 1980. The \TeX\_82 program, which was written by the author during the latter part of 1981 and the early part of 1982, also incorporates ideas from the 1979 implementation of \TeX in MESA that was written by Leonidas Guibas, Robert Sedgewick, and Douglas Wyatt at the Xerox Palo Alto Research Center. Several hundred refinements were introduced into \TeX\_82 based on the experiences gained with the original implementations, so that essentially every part of the system has been substantially improved. After the appearance of “Version 0” in September 1982, this program benefited greatly from the comments of many other people, notably David R. Fuchs and Howard W. Trickey. A final revision in September 1989 extended the input character set to eight-bit codes and introduced the ability to hyphenate words from different languages, based on some ideas of Michael J. Ferguson.

No doubt there still is plenty of room for improvement, but the author is firmly committed to keeping \TeX\_82 “frozen” from now on; stability and reliability are to be its main virtues.

On the other hand, the \texttt{WEB} description can be extended without changing the core of \TeX\_82 itself, and the program has been designed so that such extensions are not extremely difficult to make. The \texttt{banner} string defined here should be changed whenever \TeX undergoes any modifications, so that it will be clear which version of \TeX might be the guilty party when a problem arises.

If this program is changed, the resulting system should not be called \TeX; the official name \TeX by itself is reserved for software systems that are fully compatible with each other. A special test suite called the “TRIP test” is available for helping to determine whether a particular implementation deserves to be known as \TeX [cf. Stanford Computer Science report CS1027, November 1984].

Even though \TeX\_FPC does not differ from \TeX I proudly change the banner! And take responsibility for any error.

\texttt{define banner = \texttt{This is \TeX-FPC, 4th ed.}}
4* The program begins with a normal Pascal program heading, whose components will be filled in later, using the conventions of \TeX. For example, the portion of the program called ‘\{Global variables 13\}’ below will be replaced by a sequence of variable declarations that starts in §13 of this documentation. In this way, we are able to define each individual global variable when we are prepared to understand what it means; we do not have to define all of the globals at once. Cross references in §13, where it says “See also sections 20, 26, …” also make it possible to look at the set of all global variables, if desired. Similar remarks apply to the other portions of the program heading.

Actually the heading shown here is not quite normal: The program line does not mention any output file, because \TeX would ask the \TeX user to specify a file name if output were specified here.

Pascal wants the identifiers of the standard text files input and output in the parameter list of the program header.

\begin{verbatim}
define term_in \equiv i\&n\&p\&u\&t
#define term_out \equiv o\&u\&t\&p\&u\&t
define mtypenow \equiv t\&y\&\&e
format mtype \equiv type \{ ‘mtype’ will be equivalent to ‘type’ \}
format type \equiv true \{ but ‘type’ will not be treated as a reserved word \}
\end{verbatim}

(Compiler directives 9*)

\begin{verbatim}
program \textit{TEX}(term_in, term_out);
  label \{ Labels in the outer block 6 \}
  const \{ Constants in the outer block 11* \}
  mtype \{ Types in the outer block 18 \}
  var \{ Global variables 13 \}
  procedure catch_signal(i: integer); interrupt forward;
  procedure initialize; \{ this procedure gets things started properly \}
    var \{ Local variables for initialization 19 \}
    begin \{ Initialize whatever \TeX might access 8 \}
    end;
  \end{verbatim}

(Basic printing procedures 57)

(Errors handling procedures 78)

7* Some of the code below is intended to be used only when diagnosing the strange behavior that sometimes occurs when \TeX is being installed or when system wizards are fooling around with \TeX without quite knowing what they are doing. Such code will not normally be compiled; it is delimited by the codewords ‘\texttt{debug \ldots gusted}’, with apologies to people who wish to preserve the purity of English.

Similarly, there is some conditional code delimited by ‘\texttt{stat \ldots tats}’ that is intended for use when statistics are to be kept about \TeX’s memory usage. The \texttt{stat \ldots tats} code also implements diagnostic information for \texttt{\textbackslash tracingparagraphs}, \texttt{\textbackslash tracingpages}, and \texttt{\textbackslash tracingrestores}.

\begin{verbatim}
define debug \equiv @\{ \{ change this to ‘debug \equiv ’ when debugging \} 
define gusted \equiv @\{ \{ change this to ‘gusted \equiv ’ when debugging \} 
format debug \equiv begin
format gusted \equiv end
\end{verbatim}

\begin{verbatim}
define stat \equiv \{ \{ change this to ‘stat \equiv ‘ to turn off statistics \} 
define tats \equiv \{ \{ change this to ‘tats \equiv ‘ to turn off statistics \} 
format stat \equiv begin
format tats \equiv end
\end{verbatim}
9* If the first character of a Pascal comment is a dollar sign, Pascal-H treats the comment as a list of “compiler directives” that will affect the translation of this program into machine language. The directives shown below specify full checking and inclusion of the Pascal debugger when \textsc{TeX} is being debugged, but they cause range checking and other redundant code to be eliminated when the production system is being generated. Arithmetic overflow will be detected in all cases.

X If the first character of a Pascal comment is a dollar sign, Free Pascal treats the comment as a “compiler directive”. Turn off checking since the debugger might trigger a range check when it accesses subfields of a memory word without knowing what it is reading. Overflow is checked if the result of an integer operation overflows the range of 64bit \textit{integer}. FPC in default mode neither provides \texttt{goto} nor the I/O procedures \texttt{get} and \texttt{put}, and 16-bit \textit{integer}. The compiler directive \texttt{MODE ISO} fixes all of it.

\begin{verbatim}
( Compiler directives 9* )  \equiv
  @{$MODE ISO@}  \{ turn on mode ISO \}
  @{$Q@}        \{ turn on overflow checking \}
  @{$R@}        \{ turn on range checking \}
  debug @{$Q@}  @{$R@} gubed \{ turn off all checks when debugging \}
\end{verbatim}

This code is used in section 4*.

10* This \textsc{TeX} implementation conforms to the rules of the \textit{Pascal User Manual} published by Jensen and Wirth in 1975, except where system-dependent code is necessary to make a useful system program, and except in another respect where such conformity would unnecessarily obscure the meaning and clutter up the code: We assume that \texttt{case} statements may include a default case that applies if no matching label is found. Thus, we shall use constructions like

\begin{verbatim}
case \textit{x} of
  1: (code for \textit{x} = 1);
  3: (code for \textit{x} = 3);
othercases (code for \textit{x} \neq 1 and \textit{x} \neq 3)
endcases
\end{verbatim}

since most Pascal compilers have plugged this hole in the language by incorporating some sort of default mechanism. For example, the \textsc{Pascal-H} compiler allows \texttt{`others:`} as a default label, and other Pascals allow syntaxes like \texttt{`else`} or \texttt{`otherwise`} or \texttt{`otherwise:`}, etc. The definitions of \texttt{othercases} and \texttt{endcases} should be changed to agree with local conventions. Note that no semicolon appears before \texttt{endcases} in this program, so the definition of \texttt{endcases} should include a semicolon if the compiler wants one. (Of course, if no default mechanism is available, the \texttt{case} statements of \textsc{TeX} will have to be laboriously extended by listing all remaining cases. People who are stuck with such Pascals have, in fact, done this, successfully but not happily!)

X This is the only place I voluntarily use an \textsc{FPC} extension to Pascal.

\begin{verbatim}
define othercases \equiv else  \{ default for cases not listed explicitly \}
define endcases \equiv end    \{ follows the default case in an extended case statement \}
format othercases \equiv else
format endcases \equiv end
\end{verbatim}
11* The following parameters can be changed at compile time to extend or reduce \TeX 's capacity. They may have different values in INITEX and in production versions of \TeX.

\begin{itemize}
\item \texttt{mem\_max} = 30000; \{ greatest index in \TeX 's internal \texttt{mem} array; must be strictly less than \texttt{max\_halfword} \}
\item \texttt{mem\_min} = 0; \{ smallest index in \TeX 's internal \texttt{mem} array; must be \texttt{min\_halfword} or more; must be equal to \texttt{mem\_bot} in INITEX, otherwise \leq \texttt{mem\_bot} \}
\item \texttt{buf\_size} = 500; \{ maximum number of characters simultaneously present in current lines of open files and in control sequences between \texttt{\csname} and \texttt{\endcsname}; must not exceed \texttt{mar\_halfword} \}
\item \texttt{error\_line} = 72; \{ width of context lines on terminal error messages \}
\item \texttt{half\_error\_line} = 42; \{ width of first lines of contexts in terminal error messages; should be between 30 and \texttt{error\_line} \}
\item \texttt{max\_print\_line} = 79; \{ width of longest text lines output; should be at least 60 \}
\item \texttt{stack\_size} = 200; \{ maximum number of simultaneous input sources \}
\item \texttt{max\_in\_open} = 6; \{ maximum number of input files and error insertions that can be going on simultaneously \}
\item \texttt{font\_max} = 75; \{ maximum internal font number; must not exceed \texttt{max\_quarterword} and must be at most \texttt{font\_base} + 256 \}
\item \texttt{font\_mem\_size} = 20000; \{ number of words of \texttt{font\_info} for all fonts \}
\item \texttt{param\_size} = 60; \{ maximum number of simultaneous macro parameters \}
\item \texttt{nest\_size} = 40; \{ maximum number of semantic levels simultaneously active \}
\item \texttt{max\_str\_gs} = 3000; \{ maximum number of strings; must not exceed \texttt{max\_halfword} \}
\item \texttt{string\_vacancies} = 8000; \{ the minimum number of characters that should be available for the user's control sequences and font names, after \TeX 's own error messages are stored \}
\item \texttt{pool\_size} = 32000; \{ maximum number of characters in strings, including all error messages and help texts, and the names of all fonts and control sequences; must exceed \texttt{string\_vacancies} by the total length of \TeX 's own strings, which is currently about 23000 \}
\item \texttt{save\_size} = 600; \{ space for saving values outside of current group; must be at most \texttt{mar\_halfword} \}
\item \texttt{trie\_size} = 8000; \{ space for hyphenation patterns; should be larger for INITEX than it is in production versions of \TeX \}
\item \texttt{trie\_op\_size} = 500; \{ space for "opcodes" in the hyphenation patterns \}
\item \texttt{dvi\_buf\_size} = 800; \{ size of the output buffer; must be a multiple of 8 \}
\item \texttt{file\_name\_size} = 40; \{ file names shouldn't be longer than this \}
\end{itemize}

\begin{verbatim}
U pool\_name = 'Texformats/tex.pool'; \{ Unix filename. \}
\end{verbatim}

This code is used in section 4*.
The ASCII code is “standard” only to a certain extent, since many computer installations have found it advantageous to have ready access to more than 94 printing characters. Appendix C of The \TeX\book gives a complete specification of the intended correspondence between characters and \TeX\’s internal representation.

If \TeX\ is being used on a garden-variety Pascal for which only standard ASCII codes will appear in the input and output files, it doesn’t really matter what codes are specified in \texttt{xhr[0...37]}, but the safest policy is to blank everything out by using the code shown below.

However, other settings of \texttt{xhr} will make \TeX\ more friendly on computers that have an extended character set, so that users can type things like ‘#’ instead of ‘\&e’. People with extended character sets can assign codes arbitrarily, giving an \texttt{xhr} equivalent to whatever characters the users of \TeX\ are allowed to have in their input files. It is best to make the codes correspond to the intended interpretations as shown in Appendix C whenever possible; but this is not necessary. For example, in countries with an alphabet of more than 26 letters, it is usually best to map the additional letters into codes less than ‘$0$. To get the most “permissive” character set, change ‘u’ on the right of these assignment statements to \texttt{chr(i)}.

\begin{verbatim}
\{ Set initial values of key variables 21 \}
\for i ← 0 to 37 \do \texttt{xhr[i] ← \textquoteleft u\textquoteright ;}
\u for i ← 11 to 37 \do \texttt{xhr[i] ← \texttt{chr(11)}}; \{ accept horizontal tab \}
\texttt{xhr[14] ← \texttt{chr(14)}}; \{ accept form feed \}
\u for i ← 177 to 377 \do \texttt{xhr[i] ← \textquoteleft u\textquoteright ;}
\end{verbatim}
25 Input and output. The bane of portability is the fact that different operating systems treat input and output quite differently, perhaps because computer scientists have not given sufficient attention to this problem. People have felt somehow that input and output are not part of "real" programming. Well, it is true that some kinds of programming are more fun than others. With existing input/output conventions being so diverse and so messy, the only sources of joy in such parts of the code are the rare occasions when one can find a way to make the program a little less bad than it might have been. We have two choices, either to attack I/O now and get it over with, or to postpone I/O until near the end. Neither prospect is very attractive, so let's get it over with.

The basic operations we need to do are (1) inputting and outputting of text, to or from a file or the user's terminal; (2) inputting and outputting of eight-bit bytes, to or from a file; (3) instructing the operating system to initiate ("open") or to terminate ("close") input or output from a specified file; (4) testing whether the end of an input file has been reached.

\TeX needs to deal with two kinds of files. We shall use the term \textit{alpha file} for a file that contains textual data, and the term \textit{byte file} for a file that contains eight-bit binary information. These two types turn out to be the same on many computers, but sometimes there is a significant distinction, so we shall be careful to distinguish between them. Standard protocols for transferring such files from computer to computer, via high-speed networks, are now becoming available to more and more communities of users.

The program actually makes use also of a third kind of file, called a \textit{word file}, when dumping and reloading base information for its own initialization. We shall define a word file later; but it will be possible for us to specify simple operations on word files before they are defined.

\begin{verbatim}
\{Types in the outer block 18\} +≡
eight_bits = 0..255; \{ unsigned one-byte quantity \}
\textit{alpha_file} = t0x0kx0kt; \{ the type of text files is \textit{text} \}
\textit{byte_file} = \texttt{packed file of eight_bits}; \{ files that contain binary data \}
\textit{untyped_file} = \texttt{file}; \{ untyped files for buffered output \}
\end{verbatim}
27 The Pascal-H compiler with which the present version of \TeX{} was prepared has extended the rules of Pascal in a very convenient way. To open file \textit{f}, we can write

\begin{verbatim}
reset (f, name, '/0') for input;
rewrite (f, name, '/0') for output.
\end{verbatim}

The ‘name’ parameter, which is of type ‘\texttt{packed array [any] of char’}, stands for the name of the external file that is being opened for input or output. Blank spaces that might appear in \texttt{name} are ignored.

The ‘/0’ parameter tells the operating system not to issue its own error messages if something goes wrong. If a file of the specified name cannot be found, or if such a file cannot be opened for some other reason (e.g., someone may already be trying to write the same file), we will have \texttt{erstat(f) \neq 0} after an unsuccessful \texttt{reset} or \texttt{rewrite}. This allows \TeX{} to undertake appropriate corrective action.

\begin{verbatim}
X The procedure \texttt{fpc_assign} assigns an external file name to a file. The function \texttt{fpc_io_result} returns a nonzero value if any error occurred since the last invocation of \texttt{fpc_io_result}. The runtime system halts the program when it experiences an I/O error. Since \TeX{}FPC wants to survive while trying to open a nonexistence file, it turns off I/O checking for the open procedures.
\end{verbatim}

\begin{verbatim}
define fpc_io_result \equiv i\$o$r0\$s\$k\$t
#define fpc_assign \equiv a\$s\$k\$g\$n
#define reset_OK(#) \equiv fpc_io_result = 0
#define rewrite_OK(#) \equiv fpc_io_result = 0
#define clear_io_result \equiv if fpc_io_result = 0 then do nothing
@0(\$f-0) \{ turn of I/O checking \}

function a_open_in(var f : alpha_file) : boolean;
    begin clear_io_result; fpc_assign(f, name_of_file); reset(f); a_open_in \leftarrow \mathrm{reset}\_\mathrm{OK}(f);
    end;

function a_open_out(var f : alpha_file) : boolean; \{ open a text file for output \}
    begin clear_io_result; fpc_assign(f, name_of_file); rewrite(f); a_open_out \leftarrow \mathrm{rewrite}\_\mathrm{OK}(f);
    end;

function b_open_in(var f : byte_file) : boolean; \{ open a binary file for input \}
    begin clear_io_result; fpc_assign(f, name_of_file); reset(f); b_open_in \leftarrow \mathrm{reset}\_\mathrm{OK}(f);
    end;

function b_open_out(var f : byte_file) : boolean; \{ open a binary file for output \}
    begin clear_io_result; fpc_assign(f, name_of_file); rewrite(f); b_open_out \leftarrow \mathrm{rewrite}\_\mathrm{OK}(f);
    end;

function w_open_in(var f : word_file) : boolean; \{ open a word file for input \}
    begin clear_io_result; fpc_assign(f, name_of_file); reset(f); w_open_in \leftarrow \mathrm{reset}\_\mathrm{OK}(f);
    end;

function w_open_out(var f : word_file) : boolean; \{ open a word file for output \}
    begin clear_io_result; fpc_assign(f, name_of_file); rewrite(f); w_open_out \leftarrow \mathrm{rewrite}\_\mathrm{OK}(f);
    end;
@0(\$f+0) \{ turn on I/O checking \}
\end{verbatim}
31* The \texttt{input} function brings the next line of input from the specified file into available positions of the buffer array and returns the value \texttt{true}, unless the file has already been entirely read, in which case it returns \texttt{false} and sets last $\leftarrow$ first. In general, the \texttt{ASCII} code numbers that represent the next line of the file are input into \texttt{buffer[first]}, \texttt{buffer[first + 1]}, \ldots, \texttt{buffer[last − 1]}; and the global variable last is set equal to first plus the length of the line. Trailing blanks are removed from the line; thus, either last $=$ first (in which case the line was entirely blank) or \texttt{buffer[last − 1] $\neq$ "."

An overflow error is given, however, if the normal actions of \texttt{input} would make last $\geq$ \texttt{bufsize}; this is done so that other parts of \TeX can safely look at the contents of \texttt{buffer[last + 1]} without overstepping the bounds of the \texttt{buffer} array. Upon entry to \texttt{input}, the condition first $<$ \texttt{bufsize} will always hold, so that there is always room for an "empty" line.

The variable \texttt{maxbufstack}, which is used to keep track of how large the \texttt{bufsize} parameter must be to accommodate the present job, is also kept up to date by \texttt{input}.

If the \texttt{bypass_eoln} parameter is \texttt{true}, \texttt{input} will do a \texttt{get} before looking at the first character of the line; this skips over an \texttt{eoln} that was in \texttt{f}. The procedure does not do a \texttt{get} when it reaches the end of the line; therefore it can be used to acquire input from the user's terminal as well as from ordinary text files.

Standard Pascal says that a file should have \texttt{eoln} immediately before \texttt{eof}, but \TeX needs only a weaker restriction: If \texttt{eof} occurs in the middle of a line, the system function \texttt{eoln} should return a \texttt{true} result (even though \texttt{f} will be undefined).

Since the inner loop of \texttt{input} is part of \TeX's "inner loop"—each character of input comes in at this point—it is wise to reduce system overhead by making use of special routines that read in an entire array of characters at once, if such routines are available. The following code uses standard Pascal to illustrate what needs to be done, but finer tuning is often possible at well-developed Pascal sites.

P Standard Pascal never suppresses the first \texttt{get}, so \texttt{input} must not bypass the first character of the first line. To maintain this rule for subsequent lines, \texttt{input} is changed to bypass the end of line character at the end of line.

\begin{verbatim}
function input (var f : alpha_file; bypass_eoln : boolean): boolean;
  { inputs the next line or returns false }
var last_nonblank: 0..bufsize;  { last with trailing blanks removed }
begin  {input the first character of the line into \texttt{f}}
  last $\leftarrow$ first;  { cf. Matthew 19:30 }
  if \texttt{eof} then \texttt{input} $\leftarrow$ \texttt{false}
else begin last_nonblank $\leftarrow$ first;
  while $\neg$\texttt{eoln} (f) do
    begin if last $\geq$ \texttt{maxbufstack} then
      begin \texttt{maxbufstack} $\leftarrow$ last + 1;
        if \texttt{maxbufstack} = \texttt{bufsize} then (Report overflow of the input buffer, and abort 35);
       end;
      \texttt{buffer}[last] $\leftarrow$ \texttt{xord}[f]; \texttt{get}(f); incr(last);
      if \texttt{buffer}[last]$-1] $\neq$ "." then last_nonblank $\leftarrow$ last;
     end;
    last $\leftarrow$ last_nonblank; \texttt{input} $\leftarrow$ \texttt{true}; \texttt{read}(f);
   end;
end;
\end{verbatim}

32* The user's terminal acts essentially like other files of text, except that it is used both for input and for output. When the terminal is considered an input file, the file variable is called \texttt{term_in}, and when it is considered an output file the file variable is \texttt{term_out}.

P No need to declare standard input/output in standard Pascal.
Here is how to open the terminal files in Pascal-H. The `/T` switch suppresses the first `get`.

In Pascal, the standard text files are opened implicitly.

```pascal
define t_open_in ≡ do nothing  { open the terminal for text input }
define t_open_out ≡ do nothing  { open the terminal for text output }
```

Sometimes it is necessary to synchronize the input/output mixture that happens on the user’s terminal, and three system-dependent procedures are used for this purpose. The first of these, `update_terminal`, is called when we want to make sure that everything we have output to the terminal so far has actually left the computer’s internal buffers and been sent. The second, `clear_terminal`, is called when we wish to cancel any input that the user may have typed ahead (since we are about to issue an unexpected error message). The third, `wake_up_terminal`, is supposed to revive the terminal if the user has disabled it by some instruction to the operating system.

In Unix, nothing needs to be done here.

```pascal
define fpc_flush ≡ f0@l0@u0@s0@h
define update_terminal ≡ fpc_flush(term_out)  { empty the terminal output buffer }
define clear_terminal ≡ do nothing  { clear the terminal input buffer }
define wake_up_terminal ≡ do nothing  { cancel the user’s cancellation of output }
```
36* Different systems have different ways to get started. But regardless of what conventions are adopted, the routine that initializes the terminal should satisfy the following specifications:

1) It should open file term_in for input from the terminal. (The file term_out will already be open for output to the terminal.)
2) If the user has given a command line, this line should be considered the first line of terminal input. Otherwise the user should be prompted with `**', and the first line of input should be whatever is typed in response.
3) The first line of input, which might or might not be a command line, should appear in locations first to last - 1 of the buffer array.
4) The global variable loc should be set so that the character to be read next by \TeX{} is in buffer[loc]. This character should not be blank, and we should have loc < last.

(It may be necessary to prompt the user several times before a non-blank line comes in. The prompt is `**' instead of the later `*' because the meaning is slightly different: `\input' need not be typed immediately after `**'.)

X An \texttt{fpc_string} is a \texttt{packed array} \texttt{[1 .. fpc_length]} of char with varying length. The function \texttt{fpc_length(s)} returns the length of the \texttt{fpc_string s}. The function \texttt{fpc_param_count} returns the number of command line arguments less one. The function \texttt{fpc_param_str(n)} returns the n-th argument for \texttt{0 \leq n \leq fpc_param_count}.

\begin{verbatim}
F This procedure puts the command line arguments separated by spaces into buffer. Like input_in it updates last so that buffer[first .. last) will contain the command line.

define loc ← cur_input.loc_field { location of first unread character in buffer }
define fpc_string q ← s$\$h$\$1$$\$0$\$0$\$0$\$r$\$0$\$0$\$0$\$s$\$1$$\$0$\$0$\$0$\$i$$\$0$\$0$\$g$
define fpc_length q ← l$\$i$$\$c$$\$0$\$n$$\$0$$\$g$$\$0$$\$t$$\$\$h$
define fpc_param_count q ← p$\$h$$\$a$$\$0$$\$r$\$0$$\$a$$\$m$$\$0$$\$m$$\$c$$\$0$$\$0$$\$u$$\$0$$\$n$$\$t$$\$t$
define fpc_param_str q ← p$\$h$$\$a$$\$0$$\$r$\$0$$\$a$$\$m$$\$s$$\$s$$\$k$$\$r$

procedure input_command_line; { get the command line in buffer }

var argc: integer; { argument counter }
arg: fpc_string q; { argument }
cc: integer; { character counter in argument }

begin last ← first; argc ← 1;
while argc ≤ fpc_param_count do
    begin cc ← 1; arg ← fpc_param_str(argc); incr(argc);
        while cc ≤ fpc_length(arg) do
            begin if last + 1 ≥ buf.size then {Report overflow of the input buffer, and abort 35};
                if xor[arg[cc]] ≠ invalid_code then buffer[last] ← xor[arg[cc]];
                    incr(last); incr(cc)
            end;
            if (argc ≤ fpc_param_count) then
                begin buffer[last] ← "_"; incr(last) { insert a space between arguments }
            end
        end
    end
end
\end{verbatim}
The following program does the required initialization without retrieving a possible command line. The command line is treated as the first terminal line.

Tell user to end the terminal file by Control-D.

```plaintext
function init_terminal: boolean; { gets the terminal input started }
  label exit;
  begin terminal_in; input_command_in; loc ← first;
  if loc < last then
    begin init_terminal ← true; return; { first line is the command line }
    end;
  loop begin write (term_out, "**");
    if ~input_in (term_in, true) then { this shouldn't happen }
      begin write_in (term_out); init_terminal ← false; return;
      end;
    loc ← first;
    while (loc < last) ∧ (buffer[loc] = ".") do incr (loc);
    if loc < last then
      begin init_terminal ← true; return; { return unless the line was all blank }
      end;
    write_in (term_out, "Please type the name of your input file or Control-D.");
  end;
  exit: end;

51* define bad_pool(#) ≡
  begin wake_up_terminal; write_in (term_out, #); get_strings_started ← false; return;
  end

(Read the other strings from the TEX.POOL file and return true, or give an error message and return false 51*) ≡

name_of_file ← pool_name; { we needn't set name_length }
if a_open_in (pool_file) then
  begin c ← false;
  repeat {Read one string, but return false if the string memory space is getting too tight for comfort 52};
    until c;
  a_close (pool_file); get_strings_started ← true;
  end

else bad_pool("!iLc'an't read Texformats/ tex.pool.") { Unix file name }

This code is used in section 47.
53* The \texttt{WEB} operation \texttt{&} denotes the value that should be at the end of this \texttt{TEX.POOL} file; any other value means that the wrong pool file has been loaded.

\begin{verbatim}
\langle Check the pool check sum \texttt{53} * \rangle \equiv
\begin{IEEEeqnarray*}{rCl}
&\hspace{0cm} & \texttt{begin a} \leftarrow 0; \ k \leftarrow 1;
\end{IEEEeqnarray*}
\begin{IEEEeqnarray*}{rCl}
\hspace{0cm} &\texttt{loop} & \texttt{begin if} \ (\texttt{xord}[n] < "0") \lor (\texttt{xord}[n] > "9") \texttt{then}
\end{IEEEeqnarray*}
\begin{IEEEeqnarray*}{rCl}
\hspace{0cm} &\hspace{0cm} & \texttt{bad_pool(} '*!\texttt{TEX.POOL}, \texttt{check}, \texttt{sum}, \texttt{doesn't have nine digits.});
\end{IEEEeqnarray*}
\begin{IEEEeqnarray*}{rCl}
\hspace{0cm} &\hspace{0cm} & \texttt{a} \leftarrow 10 \ast \texttt{a} + \texttt{xord}[p] - "0";
\end{IEEEeqnarray*}
\begin{IEEEeqnarray*}{rCl}
\hspace{0cm} &\hspace{0cm} & \texttt{if k} = 9 \texttt{then goto done;}
\end{IEEEeqnarray*}
\begin{IEEEeqnarray*}{rCl}
\hspace{0cm} &\hspace{0cm} & \texttt{incr} (k); \texttt{read (pool file, n)};
\end{IEEEeqnarray*}
\begin{IEEEeqnarray*}{rCl}
\hspace{0cm} &\texttt{end;}
\end{IEEEeqnarray*}
\begin{IEEEeqnarray*}{rCl}
\hspace{0cm} &\texttt{done; if a} \neq \texttt{&} \texttt{ then}
\end{IEEEeqnarray*}
\begin{IEEEeqnarray*}{rCl}
\hspace{0cm} &\hspace{0cm} & \texttt{bad_pool(} '*!\texttt{TEXforms/TEX.pool, doesn't have nine digits. Not installed?});
\end{IEEEeqnarray*}
\hspace{0cm}
\end{verbatim}

\texttt{u}

\begin{verbatim}
\hspace{0cm} \{ Unix file name \}
\end{verbatim}

\begin{verbatim}
\hspace{0cm} \hspace{0cm} c \leftarrow \texttt{true};
\hspace{0cm} \texttt{end}
\end{verbatim}

This code is used in section 52.

79* Individual lines of help are recorded in the array \texttt{help.line}, which contains entries in positions 0 \ldots (\texttt{help.ptr} - 1). They should be printed in reverse order, i.e., with \texttt{help.line}[0] appearing last.

\begin{verbatim}
\langle Global variables \texttt{13} \rangle \equiv
\begin{IEEEeqnarray*}{rCl}
&\texttt{help.line: array [0..5] of str.number;} & \texttt{\{ helps for the next error \}}
\end{IEEEeqnarray*}
\begin{IEEEeqnarray*}{rCl}
&\texttt{help.ptr: 0..6;} & \texttt{\{ the number of help lines present \}}
\end{IEEEeqnarray*}
\begin{IEEEeqnarray*}{rCl}
&\texttt{use.err.help: boolean;} & \texttt{\{ should the \texttt{err} \texttt{help list be shown?} \}}
\end{IEEEeqnarray*}
\begin{IEEEeqnarray*}{rCl}
&\texttt{want.edit: boolean;} & \texttt{\{ start vi? \}}
\end{IEEEeqnarray*}
\end{verbatim}

80* \langle Set initial values of key variables \texttt{21} \rangle \equiv

\begin{verbatim}
\begin{IEEEeqnarray*}{rCl}
&\texttt{help.ptr} & \leftarrow 0; \texttt{use.err.help} & \leftarrow \texttt{false};
\end{IEEEeqnarray*}
\begin{IEEEeqnarray*}{rCl}
&\texttt{want.edit} & \leftarrow \texttt{false}; \texttt{\{ don't start ed \}}
\end{IEEEeqnarray*}
\end{verbatim}
It is desirable to provide an ‘E’ option here that gives the user an easy way to return from \TeX to the system editor, with the offending line ready to be edited. But such an extension requires some system wizardry, so the present implementation simply types out the name of the file that should be edited and the relevant line number.

There is a secret ‘D’ option available when the debugging routines haven’t been commented out.

\begin{verbatim}
Interpret code \texttt{c} and \textbf{return} if done 84\* \equiv
  case \texttt{c} of
    "0", "1", "2", "3", "4", "5", "6", "7", "8", "9": \textbf{if deletions_allowed then}
      \textbf{debug "D": begin debug help; goto continue; end; gubed}
    "E": if base_ptr > 0 then
      \textbf{begin}
      \textbf{if input_stack[base_ptr].name_field ≥ 256 then}
        \textbf{begin print\_al\_d("You want to edit file utility");}
        \textbf{slow\_print(input_stack[base_ptr].name_field); print("at line "); print_int(line);}
        \textbf{interaction ← scroll\_mode; want\_edit ← true; jump\_out;}
      \end;
    "H": \textbf{begin}
      \textbf{print the help information and goto continue 89};
    "T": \textbf{begin introduce new material from the terminal and return 87};
    "Q", "R", "S": \textbf{begin}
      \textbf{interaction ← scroll\_mode; jump\_out;}
    \end;
  othercases \textbf{do nothing}
endcases;
\end{verbatim}

This code is used in section 83.

\begin{verbatim}
Here the author of \TeX apologizes for making use of the numerical relation between "Q", "R", "S", and the desired interaction settings \texttt{batch\_mode}, \texttt{nonstop\_mode}, \texttt{scroll\_mode}.

\textbf{begin error\_count ← 0; interaction ← batch\_mode + c - "Q": print("OK, entering");

\textbf{case \texttt{c} of
E "Q": print\_esc("batchmode"); \{ don't turn off terminal now \}
"R": print\_esc("nonstopmode");
"S": print\_esc("scrollmode");
end; \{ there are no other cases \}
\textbf{print("..."}); print\_ln; update\_terminal;
  \textbf{if} \texttt{c = "Q" then decr\_selecter;}
\textbf{return}; \{ but now \}
end
\end{verbatim}

This code is used in section 84\*.
109* When \( \TeX \) “packages” a list into a box, it needs to calculate the proportionality ratio by which the glue inside the box should stretch or shrink. This calculation does not affect \( \TeX \)'s decision making, so the precise details of rounding, etc., in the glue calculation are not of critical importance for the consistency of results on different computers.

We shall use the type `glue_ratio` for such proportionality ratios. A glue ratio should take the same amount of memory as an `integer` (usually 32 bits) if it is to blend smoothly with \( \TeX \)'s other data structures. Thus `glue_ratio` should be equivalent to `short_real` in some implementations of Pascal. Alternatively, it is possible to deal with glue ratios using nothing but `fixed-point` arithmetic; see TUGboat 3,1 (March 1982), 10–27. (But the routines cited there must be modified to allow negative glue ratios.)

X In FPC Pascal the type `fpce_single` seems appropriate.

\[
\begin{align*}
define \ & \text{ `fpce_single` } \equiv \text{ `real` } \ \& \ \text{ `integer` } \, \text{ `clerk` } \\
define \ & \text{ `set_glue_ratio_zero` } (\#) \equiv \# \leftarrow 0 \quad \{ \text{ store the representation of zero ratio } \} \\
define \ & \text{ `set_glue_ratio_one` } (\#) \equiv \# \leftarrow 1.0 \quad \{ \text{ store the representation of unit ratio } \} \\
define \ & \text{ `float` } (\#) \equiv \# \quad \{ \text{ convert from `glue_ratio` to `real` } \} \\
define \ & \text{ `unfloat` } (\#) \equiv \# \quad \{ \text{ convert from `real` to type `glue_ratio` } \} \\
define \ & \text{ `float_constant` } (\#) \equiv \#.0 \quad \{ \text{ convert integer constant to `real` } \}
\end{align*}
\]

\( \langle \text{Types in the outer block 18} \rangle \) \( + \equiv \)

`glue_ratio` \( \equiv \) `fpce_single`; \{ one-word representation of a glue expansion factor in FPC Pascal \}

112* The operation of adding or subtracting `min_quarterword` occurs quite frequently in \( \TeX \), so it is convenient to abbreviate this operation by using the macros `qi` and `qo` for input and output to and from `quarterword` format.

The inner loop of \( \TeX \) will run faster with respect to compilers that don’t optimize expressions like \( `x+0` \) and \( `x-0` \), if these macros are simplified in the obvious way when `min_quarterword` \( = \) 0.

X Which is the case with FPC.

\[
\begin{align*}
define \ & \text{ `qi` } (\#) \equiv \# \quad \{ \text{ to put an `eight_bits` item into a `quarterword` } \} \\
define \ & \text{ `qo` } (\#) \equiv \# \quad \{ \text{ to take an `eight_bits` item out of a `quarterword` } \} \\
define \ & \text{ `hi` } (\#) \equiv \# \quad \{ \text{ to put a sixteen-bit item into a `halfword` } \} \\
define \ & \text{ `ho` } (\#) \equiv \# \quad \{ \text{ to take a sixteen-bit item from a `halfword` } \}
\end{align*}
\]

241* The following procedure, which is called just before \( \TeX \) initializes its input and output, establishes the initial values of the date and time. Since standard Pascal cannot provide such information, something special is needed. The program here simply assumes that suitable values appear in the global variables `sys_time`, `sys_day`, `sys_month`, and `sys_year` (which are initialized to noon on 4 July 1776, in case the implementor is careless).

X The functions `now`, `decodedate`, and `decodetime` are provided by the unit `sysutils`. The command line option `fpcc` or `Fasyutils.tex` links that unit. When FPC is in ISO mode, it does not accept declaring a unit in the source file.

\[
\begin{align*}
define \ & \text{ `fpce_now` } \equiv \text{ `now` } \\
define \ & \text{ `fpce_decode_date` } \equiv \text{ `decodedate` } \\
define \ & \text{ `fpce_decode_time` } \equiv \text{ `decodetime` }
\end{align*}
\]

procedure `fpce_decode_time(\#);` \{ current time \}

\[
\begin{align*}
\var \ & \text{ `yy`, `mm`, `dd` : `word` } \ ; \ \text{ `hh`, `ss`, `ms` : `word` } \\
\begin{begin}
define \ & \text{ `fpce_decode_date` } (\#) \equiv \text{ `decode_date` } \\
define \ & \text{ `fpce_decode_time` } (\#) \equiv \text{ `decodetime` } \\
\end{begin}
\end{align*}
\]
All of the easy branches of get\_next have now been taken care of. There is one more branch.

\texttt{TeX82} ends the current line by calling \texttt{printf} even if the line is empty. This causes a spurious ugly empty line. Calling \texttt{printnl("\n")} is smarter. It ends the current line only if it is not empty.

\begin{verbatim}
define end\_line\_char\_inactive \equiv (end\_line\_char < 0) \lor (end\_line\_char > 255)

\langle Move to next line of file, or \texttt{goto restart} if there is no next line, or \texttt{return} if a \texttt{read} line has finished \rangle

\textbf{if} \texttt{name > 17} \textbf{then}

\langle \text{Read next line of file into buffer, or \texttt{goto restart} if the file has ended} \rangle

\textbf{else} \textbf{begin if } \texttt{\neg terminal\_input} \textbf{then}

\langle \texttt{\read line has ended} \rangle

\texttt{cur\_cmd \leftarrow 0; cur\_chr \leftarrow 0; return;}

\textbf{end;}

\textbf{if} \texttt{input\_ptr > 0} \textbf{then} \{ \text{text was inserted during error recovery} \}

\texttt{begin end\_file\_reading; \texttt{goto restart}; \{ \text{resume previous level} \}}

\textbf{end;}

\textbf{if} \texttt{selector < log\_only} \textbf{then} \texttt{open log\_file;}

\textbf{if} \texttt{interaction > nonstop\_mode} \textbf{then}

\texttt{begin if} \texttt{end\_line\_char\_inactive} \texttt{then} \texttt{incr (limit);} 

\textbf{if} \texttt{limit = -1} \textbf{then} \{ \text{previous line was empty} \}

\begin{verbatim}
  print\_nl("(Please type a command or say \\textbackslash \end\"");
  print\_nl("\n"); first \leftarrow \texttt{start}; prompt\_input("\*\*\*"); \{ \text{input on-line into buffer} \}
  limit \leftarrow \texttt{last};

  \textbf{if} \texttt{end\_line\_char\_inactive} \textbf{then} \texttt{decr (limit)}

  \textbf{else} \texttt{buffer[limit] \leftarrow end\_line\_char};

  first \leftarrow limit + 1; loc \leftarrow \texttt{start};

\end{verbatim}

\texttt{end}

\textbf{else} \texttt{fatal\_error("*** (job aborted, \texttt{nonlegal} end\_found)"}; \{ \text{nonstop mode, which is intended for overnight batch processing, never waits for on-line input} \}

\textbf{end}

This code is used in section 343.

Input files that can’t be found in the user’s area may appear in a standard system area called \texttt{TEX\_area}. Font metric files whose areas are not given explicitly are assumed to appear in a standard system area called \texttt{TEX\_font\_area}. These system area names will, of course, vary from place to place.

\textbf{Use the Unix \texttt{file separator}.}

\begin{verbatim}
define \texttt{TEX\_area} \equiv \texttt{"TeXinputs/" } \{ \text{i.e., a subdirectory of the working directory} \}

define \texttt{TEX\_font\_area} \equiv \texttt{"TeXfonts/" } \{ \text{dito} \}
\end{verbatim}
516* And here’s the second. The string pool might change as the file name is being scanned, since a new `\csname` might be entered; therefore we keep `area_delimiter` and `ext_delimiter` relative to the beginning of the current string, instead of assigning an absolute address like `pool_ptr` to them.

```pascal
function more_name(c : ASCII_code) : boolean ;
    begin if c = "_" then more_name ← false
else begin str_room(1); append_char(c);  { contribute c to the current string }
        if c = "/
           then { use "\" as a file name separator }
                       begin area_delimiter ← cur_length; ext_delimiter ← 0;
        end
else if (c = ".") ∧ (ext_delimiter = 0) then ext_delimiter ← cur_length;
        more_name ← true;
end;
end;
```

519* Another system-dependent routine is needed to convert three internal T\TeX strings into the `name_of_file` value that is used to open files. The present code allows both lowercase and uppercase letters in the file name.

```pascal
In Unix strings are terminated by chr(0).

define append_to_name(#) ≡
    begin c ← #; incr(k);
        if k ≤ file_name_size then name_of_file[k] ← xchr[c];
    end

procedure pack_file_name(n, a, e : str_number);
    var k: integer;  { number of positions filled in name_of_file }
        c: ASCII_code;  { character being packed }
        j: pool_pointer;  { index into str_pool }
    begin k ← 0;
        for j ← str_start[a] to str_start[a + 1] - 1 do append_to_name(so(str_pool[j]));
        for j ← str_start[n] to str_start[n + 1] - 1 do append_to_name(so(str_pool[j]));
        for j ← str_start[e] to str_start[e + 1] - 1 do append_to_name(so(str_pool[j]));
        if k ≤ file_name_size then name_length ← k else name_length ← file_name_size;
        for k ← name_length + 1 to file_name_size do name_of_file[k] ← chr(0);
    end;
```

521* ⟨Set initial values of key variables 21⟩ ≡

```pascal
TEX_format_default ← 'Texforms/plain.fmt';  { "/" is the Unix file name separator }
```
523* Here is the messy routine that was just mentioned. It sets name_of_file from the first n
characters of TEX_format_default, followed by buffer[a . . b], followed by the last format_ext_length
characters of TEX_format_default.

We dare not give error messages here, since TeX calls this routine before the error routine
is ready to roll. Instead, we simply drop excess characters, since the error will be detected in
another way when a strange file name isn't found.

procedure pack_buffered_name (n : small_number; a, b : integer);
var k : integer; { number of positions filled in name_of_file }
c : ASCII_code; { character being packed }
j : integer; { index into buffer or TEX_format_default }
begin if n + b - a + 1 + format_ext_length > file_name_size then
  b ← a + file_name_size - n - 1 - format_ext_length;
k ← 0;
for j ← 1 to n do append_to_name (xord [TEX_format_default [j]]);
for j ← a to b do append_to_name (buffer [j]);
for j ← format_default_length - format_ext_length + 1 to format_default_length do
  append_to_name (xord [TEX_format_default [j]]);
if k ≤ file_name_size then name_length ← k else name_length ← file_name_size;
for k ← name_length + 1 to file_name_size do name_of_file [k] ← chr (0);
end;

524* Here is the only place we use pack_buffered_name. This part of the program becomes active
when a "virgin" TeX is trying to get going, just after the preliminary initialization, or when the
user is substituting another format file by typing & after the initial ‘**’ prompt. The buffer
contains the first line of input in buffer[loc . . (last - 1)], where loc < last and buffer[loc] ≠ "u".

(Declare the function called open_fmt_file 524*)

function open_fmt_file : boolean;
  label found, exit;
var j : 0 . . buf_size; { the first space after the format file name }
begin j ← loc;
if buffer[loc] = "&" then
  begin incr (loc); j ← loc; buffer [last] ← "u";
   while buffer [j] ≠ "u" do incr (j);
pack_buffered_name (0, loc, j - 1); { try first without the system file area }
if w_open_in (fmt_file) then goto found;
pack_buffered_name (format_area_length, loc, j - 1); { now try the system format file area }
if w_open_in (fmt_file) then goto found;
  wake_up_terminal;
  wterm_h ("Sorry, I can’t find that format; I will try plain.");
  update_terminal;
  end; { now pull out all the stops: try for the system plain file }
pack_buffered_name (format_default_length - format_ext_length, 1, 0);
if ¬w_open_in (fmt_file) then
  begin wake_up_terminal; wterm_h ("I can’t find TeX formats/plain.fmt!");
       { Unix file name }
open_fmt_file ← false; return;
  end;
found : loc ← j; open_fmt_file ← true;
exit : end;

This code is used in section 1303.
530 If some trouble arises when \TeX{} tries to open a file, the following routine calls upon the user to supply another file name. Parameter $s$ is used in the error message to identify the type of file; parameter $e$ is the default extension if none is given. Upon exit from the routine, variables \texttt{cur\_name}, \texttt{cur\_area}, \texttt{cur\_ext}, and \texttt{name\_of\_file} are ready for another attempt at file opening.

\begin{verbatim}
procedure prompt\_file\_name (s, e : str\_number);

label done;
var k : 0 . . buf\_size; \{ index into buffer \}
begin if interaction = scroll\_mode then wake\_up\_terminal;
if s = "input\_file\_name" then print\_err ("I\_can\_t find\_file\_\"");
else print\_err ("I\_can\_t write\_\"");
print\_file\_name (cur\_name, cur\_area, cur\_ext); print (".");
if e = ".tex" then show\_context;
print\_nl ("Please type another"); print\_nl ("or Control-D");
if interaction < scroll\_mode then
  fatal\_error ("*** job\_aborted, file\_error\_in\_nonstop\_mode");
clear\_terminal; prompt\_input (";"); \{ Scan file name in the buffer 531 \};
if cur\_ext = "" then cur\_ext ← e;
pack\_cur\_name;
end;
\end{verbatim}
537* Let's turn now to the procedure that is used to initiate file reading when an `\input` command is being processed. Beware: For historic reasons, this code foolishly conserves a tiny bit of string pool space; but that can confuse the interactive `\E` option.

In fact, it breaks the `\E` option whenever the file to be edited was opened after the log file. In that case, the last string constructed is the name of the log file, otherwise, the last string constructed is the name of the input file. If the name of the input file is the last string constructed, \TeX\ strips off area and extension to conserve string pool space. The user is shown the base name of the file he wants to edit to fix a bug.

Sadly, Knuth doesn't dare to fix this bug, which is known for at least twelve years. Are we approaching the limits of “Literate Programming”. It looks beautiful but does it really help to cope with complexity?

\begin{verbatim}
procedure start_input; \{ \TeX \ will \ \texttt{\input} \ something \}
label done;
begin scan_file_name; \{ set cur_name to desired file name \}
if cur_ext = "" then cur_ext \leftarrow \".tex\";
pack_cur_name;
loop begin begin_file_reading; \{ set up cur_file and new level of input \}
if a_open_in (cur_file) then goto done;
if cur_area = "" then
begin pack_file_name (cur_name, TEX_area, cur_ext);
if a_open_in (cur_file) then goto done;
end;
end_file_reading; \{ remove the level that didn't work \}
prompt_file_name("input_{file_name}"", \".tex\");
end;

done: name \leftarrow a_make_name_string (cur_file);
if job_name = 0 then
begin job_name \leftarrow cur_name; open_log_file;
end; \{ open_log_file doesn't show context, so limit and loc needn't be set to meaningful values yet \}
if term_offset + length (name) > max_print_line - 2 then print_In
else if (term_offset > 0) \lor (file_offset > 0) then print_char (\"\l")
print_char (\"\"); incr (open_paren); slow_print (name); update_terminal; state \leftarrow new_line;
\langle Read the first line of the new file 538 \rangle;
end;
\end{verbatim}

575* We check to see that the \TeX\ file doesn't end prematurely; but no error message is given for files having more than \texttt{if} words.

\begin{verbatim}
\langle \text{Read font parameters 575*} \rangle \equiv
begin for \ k \leftarrow 1 \ to \ np \ do
if \ k = 1 \ then \ \{ \text{the slant parameter is a pure number} \}
begin fget; \sw \leftarrow \texttt{fbyte};
if \sw > 127 \ then \ \sw \leftarrow \sw - 256;
fget; \sw \leftarrow \sw \ast \texttt{.4} \ + \texttt{fbyte}; fget; \sw \leftarrow \sw \ast \texttt{.4} \ + \texttt{fbyte}; fget;
font_info [param_base [f]] . sc \leftarrow (\sw \ast 20) \ + (\texttt{fbyte} \ div 20);
end
else store_scaled (font_info [param_base [f] + k - 1] . sc);
for \ k \leftarrow np + 1 \ to \ 7 \ do \ font_info [param_base [f] + k - 1] . sc \leftarrow 0;
end
\end{verbatim}

This code is used in section 562.
597* The actual output of `dvi_buf[a . b]` to `dvi_file` is performed by calling `write_dvi(a, b)`. For best results, this procedure should be optimized to run as fast as possible on each particular system, since it is part of \TeX's inner loop. It is safe to assume that `a` and `b + 1` will both be multiples of 4 when `write_dvi(a, b)` is called; therefore it is possible on many machines to use efficient methods to pack four bytes per word and to output an array of words with one system call.

The procedure `fp_c_blockwrite` takes a file, the first byte in a buffer and the number of bytes to be written as parameters and writes all bytes with one system call. This fact speeds up \TeXFPC.

\begin{verbatim}
define fp_c_blockwrite \equiv b@&l@&o@&c@&k@&w@&r@&i@&t\end{verbatim}

\begin{verbatim}
procedure write_dvi(a, b; dvi_index);
begin fp_c_blockwrite(dvi_file; dvi_buf[a . b - a + 1]);
end;
\end{verbatim}

816* The first task is to move the list from `head` to `temp_head` and go into the enclosing semantic level. We also append the `\parfillskip` glue to the end of the paragraph, removing a space (or other glue node) if it was there, since spaces usually precede blank lines and instances of `"$`.

This code assumes that a `glue_node` and a `penalty_node` occupy the same number of mem words.

\begin{verbatim}
T\TeX{82} prunes discardable nodes from the beginning of a new line until it reaches a nondiscardable node. Now, if the last line of a paragraph contains discardables only, the `\parfillskip` glue at the end of the paragraph will also be removed, since it is a discardable. This will give you an empty `\hbox`. Finally \TeX appends `\rightskip` glue. This gives you a nonempty `\hbox`, raising a `Underfull` `\hbox` warning.

To avoid this happening, \TeXFPC saves a pointer to the node immediately preceding the `\parfillskip` node and quits pruning when it encounters this node several procedures later.

\begin{verbatim}
\langle Get ready to start line breaking 816* \rangle \equiv
link(temp_head) \leftarrow link(head);
if is_char_node(tail) then tail_append(new_penalty(inf_penalty))
else if type(tail) \neq glue_node then tail_append(new_penalty(inf_penalty))
else begin
  type(tail) \leftarrow penalty_node;
  delete_glue_ref(glue_ptr(tail));
  flush_node_list(leader_ptr(tail));
  penalty(tail) \leftarrow inf_penalty;
end;
non_prunable_p \leftarrow tail; \{ points to the node immediately before \parfillskip \}
link(tail) \leftarrow new_param_glue(parfillskip_code); init_char_angle \leftarrow prev_graf mod 200000;
inital_hyf \leftarrow prev_graf div 2000000; init_r_hyf \leftarrow (prev_graf div 200000) mod 100;
p_prest;
\end{verbatim}

See also sections 827, 834, and 848.

This code is used in section 815.
Breaking paragraphs into lines, continued. So far we have gotten a little way into the \texttt{line\_break} routine, having covered its important \texttt{try\_break} subroutine. Now let’s consider the rest of the process.

The main loop of \texttt{line\_break} traverses the given hlist, starting at \texttt{link(temp\_head)}, and calls \texttt{try\_break} at each legal breakpoint. A variable called \texttt{auto\_breaking} is set to true except within math formulas, since glue nodes are not legal breakpoints when they appear in formulas.

The current node of interest in the hlist is pointed to by \texttt{cur\_p}. Another variable, \texttt{prev\_p}, is usually one step behind \texttt{cur\_p}, but the real meaning of \texttt{prev\_p} is this: If \texttt{type(cur\_p) = glue\_node} then \texttt{cur\_p} is a legal breakpoint if and only if \texttt{auto\_breaking} is true and \texttt{prev\_p} does not point to a glue node, penalty node, explicit kern node, or math node.

The following declarations provide for a few other local variables that are used in special calculations.

\texttt{non\_prunable\_p} pointer.

\texttt{auto\_breaking: boolean; }{\textit{is node cur\_p outside a formula?}}
\texttt{non\_prunable\_p: pointer; }{\textit{pointer to the node before \texttt{parfillskip}}}
\texttt{prev\_p: pointer; }{\textit{helps to determine when glue nodes are breakpoints}}
\texttt{q, r, s, prev\_s: pointer; }{\textit{miscellaneous nodes of temporary interest}}
\texttt{f: internal\_font\_number; }{\textit{used when calculating character widths}}

See also section 893.

This code is used in section 815.

Once the best sequence of breakpoints has been found (hurray), we call on the procedure \texttt{post\_line\_break} to finish the remainder of the work. (By introducing this subprocedure, we are able to keep \texttt{line\_break} from getting extremely long.)

Pass \texttt{non\_prunable\_p} to the \texttt{post\_line\_break} procedure.

\texttt{post\_line\_break(final\_widow\_penalty, non\_prunable\_p)}

This code is used in section 815.
The total number of lines that will be set by \texttt{postLineBreak} is \texttt{bestLine} - \texttt{prevGraf} - 1. The last breakpoint is specified by \texttt{breakNode(best)} }, and this passive node points to the other breakpoints via the \texttt{prevBreak} links. The finishing-up phase starts by linking the relevant passive nodes in forward order, changing \texttt{prevBreak} to \texttt{nextBreak}. (The \texttt{nextBreak} fields actually reside in the same memory space as the \texttt{prevBreak} fields did, but we give them a new name because of their new significance.) Then the lines are justified, one by one.

\textbf{E} Declare another parameter. It holds the pointer to the node immediately preceding \texttt{\parfillskip}

\begin{verbatim}
define nextBreak \equiv prevBreak \{ new name for prevBreak after links are reversed \}
\end{verbatim}

\begin{verbatim}
procedure postLineBreak (finalWidowPenalty : integer; non_prunable : pointer);

\hspace{1em} label done, done1;

\hspace{1em} var q, r, s : pointer; \{ temporary registers for list manipulation \}

\hspace{1em} discBreak : boolean; \{ was the current break at a discretionary node? \}

\hspace{1em} post_discBreak : boolean; \{ and did it have a nonempty post-break part? \}

\hspace{1em} cur_width : scaled; \{ width of line number curLine \}

\hspace{1em} cur_indent : scaled; \{ left margin of line number curLine \}

\hspace{1em} t : quarterword; \{ used for replacement counts in discretionary nodes \}

\hspace{1em} pen : integer; \{ use when calculating penalties between lines \}

\hspace{1em} cur_line : halfword; \{ the current line number being justified \}

\hspace{1em} begin (Reverse the links of the relevant passive nodes, setting cur_p to the first

\hspace{1em} breakpoint 878\};

\hspace{1em} cur_line \leftarrow prev_graf + 1;

\hspace{1em} repeat (Justify the line ending at breakpoint cur_p, and append it to the current vertical

\hspace{1em} list, together with associated penalties and other insertions 880\};

\hspace{1em} incr (cur_line); cur_p \leftarrow nextBreak (cur_p);

\hspace{1em} if cur_p \neq null then

\hspace{1em} \hspace{1em} if \neg post_discBreak \texttt{ then } \langle \text{Prune unwanted nodes at the beginning of the next line 879*} \rangle;

\hspace{1em} until cur_p = null;

\hspace{1em} if (cur_line \neq bestLine) \lor (link (temp_head) \neq null) \texttt{ then } confusion("line\_breaking");

\hspace{1em} prev_graf \leftarrow bestLine - 1;

\hspace{1em} end;
\end{verbatim}
Glue and penalty and kern and math nodes are deleted at the beginning of a line, except in the anomalous case that the node to be deleted is actually one of the chosen breakpoints. Otherwise the pruning done here is designed to match the lookahead computation in `try_break`, where the `break_width` values are computed for non-discretionary breakpoints.

The pointer `non_prunable_p` references the node immediately preceding the `\parfillskip` node at the end of the paragraph. Stop pruning at this node.

```plaintext
begin r ← temp_head;
loop begin q ← link(r);
   if q = cur_break(cur_p) then goto done1; { cur_break(cur_p) is the next breakpoint }
      { now q cannot be null }
   if is_char_node(q) then goto done1;
   if non_discardable(q) then goto done1;
   if q = non_prunable_p then goto done1; { retain \parfillskip glue }
   if type(q) = kern_node then
     if subtype(q) ≠ explicit then goto done1;
   r ← q; { now type(q) = glue_node, kern_node, math_node, or penalty_node }
   end;
done1: if r ≠ temp_head then
   begin link(r) ← null; flush_node_list(link(temp_head)); link(temp_head) ← q;
   end;
end
```

This code is used in section 879*.

(Declare subprocedures for `prefixed_command` 1215 ) +≡

```plaintext
procedure new_interaction;
   begin print_nl("\n"); { print new line only if current line not empty }
      interaction ← cur_chr; {Initialize the print selector based on interaction 75};
      if log_opened then selector ← selector + 2;
   end;
```

(Undump a couple more things and the closing check word 1327* ) ≡

undump(batch_mode)(error_stop_mode)(interaction); undump(0)(str_ptr)(format_ident);
undump_int(x);
   if (x ≠ 69069) then goto bad_fmt
```

This code is used in section 1303.
1332f Now this is really it: \TeX{} starts and ends here.

The initial test involving \texttt{ready\_already} should be deleted if the Pascal runtime system is smart enough to detect such a \textquotedblleft mistake\textquotedblright.

The procedure \texttt{fp\_halt} terminates the program and passes its parameter to the shell.

\begin{verbatim}
define fp\_halt \equiv h@\&a@\&l@t
begin { start\_here }
history \leftarrow fatal\_error\_stop; \{ in case we quit during initialization \}
t\_open\_out; \{ open the terminal for output \}
if ready\_already = 314159 then goto start\_of\_TEX;
\{ Check the \textquotedblleft constant\textquotedblright\ values for consistency \}
if bad > 0 then
begin wterm\_ln(\texttt{\textup{Ouch---my internal constants have been clobbered!}}, \texttt{---case u},
bad : 1); goto final\_end;
end;
initialize; \{ set global variables to their starting values \}
init if \neg get\_strings\_started then goto final\_end;
init\_prim; \{ call primitive for each primitive \}
init\_str\_ptr \leftarrow str\_ptr; init\_pool\_ptr \leftarrow pool\_ptr; fix\_date\_and\_time;
tini
ready\_already \leftarrow 314159;
start\_of\_TEX; \{ Initialize the output routines \};
\{ Get the first line of input and prepare to start \texttt{1337} \};
history \leftarrow spotless; \{ ready to go! \}
main\_control; \{ come to life \}
final\_cleanup; \{ prepare for death \}
end\_of\_TEX; close\_files\_and\_terminate;
\end{verbatim}

\begin{verbatim}
final\_end: if want\_edit then exec\_editor; \{ user typed \texttt{E} \}
u fp\_halt (history); \{ pass history as the exit value to the system \}
end.
\end{verbatim}
Here we do whatever is needed to complete \TeX's job gracefully on the local operating system. The code here might come into play after a fatal error; it must therefore consist entirely of “safe” operations that cannot produce error messages. For example, it would be a mistake to call \texttt{str_room} or \texttt{make_string} at this time, because a call on overflow might lead to an infinite loop. (Actually there's one way to get error messages, via \texttt{prepare_mag}; but that can't cause infinite recursion.)

If \texttt{final_cleanup} is bypassed, this program doesn't bother to close the input files that may still be open. Terminate the last line on the terminal.

\begin{quote}
Last-minute procedures 1333* )

\begin{verbatim}
procedure close_files_and_terminate;

var k: integer; { all-purpose index }

begin {Finish the extensions 1378} ;
  newline_char ← -1;
  stat if tracing_stats > 0 then { Output statistics about this job 1334} ;
    tats
    wake_up_terminal ; { Finish the DVI file 642} ;
  if log_opened then
    begin wlog_cr; a_close(log_file); selector ← selector - 2;
      if selector = term_only then
        begin print_nl("Transcript written on "); slow_print(log_name); print_char(".");
          print_ln;
        end;
      end;
    end;
end;
\end{verbatim}
\end{quote}

See also sections 1335, 1336, 1338*, and 1380*.

This code is used in section 1330.
Once \TeX is working, you should be able to diagnose most errors with the `\show` commands and other diagnostic features. But for the initial stages of debugging, and for the revelation of really deep mysteries, you can compile \TeX with a few more aids, including the Pascal runtime checks and its debugger. An additional routine called `debug_help` will also come into play when you type `D` after an error message; `debug_help` also occurs just before a fatal error causes \TeX to succumb.

The interface to `debug_help` is primitive, but it is good enough when used with a Pascal debugger that allows you to set breakpoints and to read variables and change their values. After getting the prompt `debug #`, you type either a negative number (this exits `debug_help`), or zero (this goes to a location where you can set a breakpoint, thereby entering into dialog with the Pascal debugger), or a positive number \(m\) followed by an argument \(n\). The meaning of \(m\) and \(n\) will be clear from the program below. (If \(m = 13\), there is an additional argument, \(l\).)

A Pascal program must not read from the standard text file if the end of file is reached. Even in this respect, Unix and Pascal treat terminals and disk files alike.

\begin{verbatim}
define breakpoint = 888 \{ place where a breakpoint is desirable \}

\langle Last-minute procedures \rangle \equiv

def debug procedure debug_help \{ routine to display various things \}
ldef label breakpoint, exit;
vdef var k, l, m, n: integer;
begin clear_terminal;
loop
  begin wake_up_terminal; printfnl("debug\#\{\langle-1,0,exit\rangle\}"); update_terminal;
    if eof(term_in) then return \{ never read at eof \}
    read(term_in, m);
    if m < 0 then return
    else if m = 0 then
      begin goto breakpoint;
      \{ go to every declared label at least once \}
      breakpoint: m \leftarrow 0; printf\{\langle BREAKPOINT\rangle\}
    end
    else begin if eof(term_in) then return \{ never read at eof \}
      read(term_in, n);
      case m of
        (Numbered cases for debug_help \langle 1339\rangle)
        othercases print\{\langle ?\rangle\}
      endcases;
    end;
  exit: end;
gubed
\end{verbatim}
(Numbered cases for `debug help 1339`) \(\equiv\)

1. `print_word(mem[n])`; \{ display `mem[n]` in all forms \}
2. `print_int(info(n));`
3. `print_int(link(n));`
4. `print_word(eqth[n]);`
5. `print_word(font_info[n]);`
6. `print_word(save_stack[n]);`
7. `show_box(n)`; \{ show a box, abbreviated by `show_box_depth` and `show_box_breadth` \}
8. `begin breadth_max \leftarrow 10000; depth_threshold \leftarrow pool_size - pool_ptr - 10; show_node_list(n);` \{ show a box in its entirety \}
   `end;`
9. `show_token_list(n, null, 1000);`
10. `slow_print(n);`
11. `check_mem(n > 0);` \{ check wellformedness; print new busy locations if \(n > 0\) \}
12. `search_mem(n);` \{ look for pointers to \(n\) \}

\textbf{P}

13. `begin if eof(term_in) then return;` \{ never read at `eof` \}

   `read(term_in, l); printcmdchr(n, l);`

   `end;`
14. `for k \leftarrow 0 to n do print(buffer[k]);`
15. `begin font_in_short_display \leftarrow null_font; short_display(n);`

   `end;`
16. `panicking \leftarrow \neg panicking;`

This code is used in section 1338*.
1379* System-dependent changes. This section should be replaced, if necessary, by any special modifications of the program that are necessary to make \TeX{} work at a particular installation. It is usually best to design your change file so that all changes to previous sections preserve the section numbering; then everybody's version will be consistent with the published program. More extensive changes, which introduce new sections, can be inserted here; then only the index itself will get a new section number.
\textbf{F 1380*} If the user typed \texttt{E} to edit a file after confronted with an error message, \TeX will clean up and then call \texttt{edit} as its last feat.

This procedure must not print error messages, since all files are already closed.

Beware of using any \texttt{MV} 8 strings like "\texttt{vi +}" since that would change the string pool file and you'll need to rebuild all format files with the new string pool in case you disagree which editor is the system editor.

An overflow of \texttt{name_of_file} cannot happen, since \texttt{name_of_file} kept the file name while the file was being opened. The procedure \texttt{exec_edit} starts \texttt{vi} passing line number and file name.

\textbf{X} This procedure executes the Unix system editor, which is \texttt{ed} of course. In case you disagree, modify all four definitions of \texttt{ed, de, vi, iv} to select code that executes \texttt{vi} instead. This not just changes the name of the system editor, but it adds one argument that contains the line number to the argument vector. The argument vector for the system editor has two entries:

\begin{verbatim}
  ed file-name
\end{verbatim}

And the argument vector for the west coast editor three:

\begin{verbatim}
  vi +line file-name
\end{verbatim}

The system call \texttt{fp\_fp\_exec\_vp} expects two parameters, namely the name of the editor to be loaded and the argument vector, an \texttt{array} of the arguments to be passed to the editor. Unix replaces the code of \TeX by the code of the editor without forking a new process. On success this procedure does not return.

The type \texttt{fp\_char} is a pointer to a character. An argument is a null-terminated \texttt{packed array of char}. The \texttt{@}-operator applied to an argument evaluates to the address of the first entry, i.e. a pointer to a character.

The function \texttt{fp\_fp\_exec\_vp} wants the argument vector to be passed as a pointer to a pointer to a character. An \texttt{array} parameter is always passed as the address of its first entry. Therefore we must not apply the \texttt{@}-operator to the parameters of \texttt{fp\_fp\_exec\_vp}.

Note that the name of the binary is passed twice, namely as the first parameter and as the first entry of the second argument. \texttt{fp\_fp\_exec\_vp} searches for the binary in the \texttt{PATH}. And then it seems to replace \texttt{argv[0]} by the full name of the editor. At least this is what \texttt{ps -f} shows and might be a bug.

The procedure \texttt{fp\_fp\_exec\_vp} is provided by the \texttt{unit unix}. The command line option \texttt{-Faunix} links to that unit.

\begin{verbatim}
define fp\_fp\_exec\_vp \equiv fp\_exec\_vp
define fp\_char \equiv pchar
define edit\_file \equiv input\_stack [base\_ptr].name\_field
define vi \equiv \{ change this to 'vi \equiv ' when you think vi is the system editor \}
define iv \equiv \{ change this to 'iv \equiv ' when you think vi is the system editor \}
format vi \equiv begin
format iv \equiv end
define ed \equiv 0{ \{ change this to 'ed \equiv 0' when you think vi is the system editor \}
define de \equiv 0{ \{ change this to 'de \equiv 0' when you think vi is the system editor \}
format ed \equiv begin
format de \equiv end
\end{verbatim}

\texttt{\langle Last-minute procedures 1333* \rangle +\equiv}

\textbf{procedure exec_editor;}

\begin{verbatim}
const arg\_size = 100; \{ maximal size of each of the arguments \}
vi editor = 'vi'; \{ name of the binary to be started \}
iv
ed editor = 'ed'; \{ name of the binary to be started \}
de
editor\_length = 2; \{ length of the name \}
\end{verbatim}
X 1381* A signal handler is a procedure that takes one integer parameter. The procedure
\texttt{fpc.fp.signal} takes two parameters, an integer and a signal handler. The integer is the number of
the signal. When the program receives a signal with the designated number, the signal handler
gets invoked.

The integer \texttt{fpc.SIGINT} is the number of the interrupt signal. The system interrupts the
program, when the user types "C.

If \texttt{fpc.fp.get_errno} returns an integer that is not zero, an error occurred.

The identifier \texttt{fpc.signal_handler} denotes the \texttt{type} of a pointer to a signal handler. Since this
is foreign to Pascal, we use the \texttt{type cast} to \texttt{fpc.signal_handler} as a kludge.

The functions related to installing a signal handler are provided by the \texttt{unit baseunix}. The
command line option \texttt{-Fabaseunix} links to that unit.

\begin{verbatim}
define fpc.signal_handler \equiv signal\&handler
define fpc.fp.signal \equiv fp\signal
define fpc.SIGINT \equiv SIGINT
define fpc.fp.get_errno \equiv f&fp\get\errno
\end{verbatim}

\begin{verbatim}
(\texttt{Set initial values of key variables 21}) +
def fpc.fp.signal (fpc.SIGINT, fpc.signal_handler (catch_signal));
if fpc.fp.get_errno \neq 0 then
  write\ln (\texttt{Could not install signal handler: \texttt{fpc.fp.get_errno});
\end{verbatim}

\var i,l: integer; \{index into args\}
\var j: pool_pointer; \{index into str\_pool\}
\var s: str\_number; \{string to hold line number\}
\var sel: integer; \{save selector\}

\texttt{editor\_arg line\_arg file\_arg: array [1..arg\_size] of char}; \{arguments\}
\texttt{arg\_v: array [0..3] of fp\_char}; \{vector of arguments\}

\texttt{begin l \leftarrow editor\_length;}
\texttt{for j \leftarrow 1 to l do editor\_arg[j] \leftarrow editor[j];}
\texttt{editor\_arg[l + 1] \leftarrow chr(0);}
\texttt{sel \leftarrow selector; selector \leftarrow new_string; print\_int(line); selector \leftarrow sel; s \leftarrow make\_string; line\_arg[1] \leftarrow \texttt{^+}; j \leftarrow str\_start[s]; l \leftarrow length(s) + 1;}
\texttt{for i \leftarrow 2 to l do}
  \texttt{begin line\_arg[i] \leftarrow chr[str\_pool[j]]; incr(j)}
  \texttt{end};
\texttt{line\_arg[l + 1] \leftarrow chr(0);}
\texttt{j \leftarrow str\_start[edit\_file]; l \leftarrow length(edit\_file);}
\texttt{if l + 1 > arg\_size then}
  \texttt{begin write\ln (\texttt{'File\_name\_longer\_than\_100\_bytes!_Nice\_try!'}); halt(100);}
  \texttt{end;}
\texttt{for i \leftarrow 1 to l do}
  \texttt{begin file\_arg[i] \leftarrow chr[str\_pool[j]]; incr(j)}
  \texttt{end;}
\texttt{file\_arg[l + 1] \leftarrow chr(0);}
\texttt{argv[0] \leftarrow \#editor\_arg;}
\texttt{argv[1] \leftarrow \#line\_arg; argv[2] \leftarrow \#file\_arg; argv[3] \leftarrow \texttt{nil};}
\texttt{argv[1] \leftarrow \#file\_arg; argv[2] \leftarrow \texttt{nil};}
\texttt{fpc.fp.exec\_vp(editor, argv); write\ln (\texttt{'Sorry, executing\_the\_editor\_failed.'});}
\texttt{end;}

\X 1381* A signal handler is a procedure that takes one integer parameter. The procedure
\texttt{fpc.fp.signal} takes two parameters, an integer and a signal handler. The integer is the number of
the signal. When the program receives a signal with the designated number, the signal handler
gets invoked.

The integer \texttt{fpc.SIGINT} is the number of the interrupt signal. The system interrupts the
program, when the user types "C.

If \texttt{fpc.fp.get_errno} returns an integer that is not zero, an error occurred.

The identifier \texttt{fpc.signal_handler} denotes the \texttt{type} of a pointer to a signal handler. Since this
is foreign to Pascal, we use the \texttt{type cast} to \texttt{fpc.signal_handler} as a kludge.

The functions related to installing a signal handler are provided by the \texttt{unit baseunix}. The
command line option \texttt{-Fabaseunix} links to that unit.

\begin{verbatim}
define fpc.signal_handler \equiv signal\&handler
define fpc.fp.signal \equiv fp\signal
define fpc.SIGINT \equiv SIGINT
define fpc.fp.get_errno \equiv f&fp\get\errno
\end{verbatim}

\begin{verbatim}
(\texttt{Set initial values of key variables 21}) +
def fpc.fp.signal (fpc.SIGINT, fpc.signal_handler (catch_signal));
if fpc.fp.get_errno \neq 0 then
  write\ln (\texttt{Could not install signal handler: \texttt{fpc.fp.get_errno});
\end{verbatim}
The signal handler has the modifier `interrupt`. Modifiers are an extension of FPC Pascal. This one makes the compiler generate code suitable for a signal handler, which must not return to the caller, i.e., the system, but to the instruction where the program was interrupted.

```pascal
procedure catch_signal; interrupt;
    begin interrupt ← i;
    end;
```
\section*{Index}

Here is where you can find all uses of each identifier in the program, with underlined entries pointing to where the identifier was defined. If the identifier is only one letter long, however, you get to see only the underlined entries. All references are to section numbers instead of page numbers.

This index also lists error messages and other aspects of the program that you might want to look up some day. For example, the entry for “system dependencies” lists all sections that should receive special attention from people who are installing \TeX in a new operating environment. A list of various things that can’t happen appears under “this can’t happen”. Approximately 40 sections are listed under “inner loop”; these account for about 60\% of \TeX’s running time, exclusive of input and output.

The following sections were changed by the change file: 2, 4, 7, 9, 10, 11, 23, 25, 27, 31, 32, 33, 34, 36, 37, 51, 53, 79, 80, 84, 86, 109, 112, 121, 300, 514, 516, 519, 521, 523, 524, 530, 537, 575, 597, 816, 862, 876, 877, 879, 1205, 1327, 1332, 1333, 1338, 1339, 1379, 1380, 1381, 1382, 1383.

\begin{itemize}
\item *: 37, 354.
\item *: 174, 176, 178, 313, 360*556, 1006, 1355.
\item \texttt{\verb|-|}: 294.
\item \texttt{\textasciitilde}: 363.
\item ???: 59.
\item ??: 83.
\item \texttt{\textasciitilde}: 856.
\item \texttt{\textasciitilde}: 846.
\item \texttt{T\textbackslash{}X-Bug: 816*}
\item \texttt{-baseunix-: 1381*}
\item \texttt{-bypass eoln-: 31*}
\item \texttt{-unit unix-: 1380*}
\item a: 47, 102, 218, 518, 519*523*560, 597*, 691, 722, 738, 752, 1123, 1194, 1211, 1236, 1257.
\item A \texttt{<box>} was supposed to...: 1084.
\item \texttt{a\textbackslash{}close: 28, 51*329, 485, 486, 1275, 1333*1374, 1378}.
\item \texttt{a\_leaders: 149, 189, 625, 627, 634, 636, 656, 671, 1071, 1072, 1073, 1078, 1148}.
\item \texttt{a\_make\_name\_string: 525, 534, 537*}
\item \texttt{a\_open\_in: 27*51*537*1275}.
\item \texttt{a\_open\_out: 27*534, 1374}.
\item \texttt{A\_token: 445}.
\item \texttt{abort: 560, 563, 564, 565, 568, 569, 570, 571, 573}.
\item \texttt{above: 208, 1046, 1178, 1179, 1180}.
\item \texttt{\textbackslash{}above\_primitive: 1178}.
\item \texttt{above\_code: 1178, 1179, 1182, 1183}.
\item \texttt{above\_display\_short\_skip: 224, 814}.
\item \texttt{\textbackslash{}abovedisplayshortskip\_primitive: 226}.
\item \texttt{above\_display\_short\_skip\_primitive: 226}.
\item \texttt{above\_display\_short\_skip\_primitive\_code: 224, 225, 226, 1203}.
\item \texttt{above\_display\_skip: 224, 814}.
\item \texttt{\textbackslash{}abovedisplayskip\_primitive: 226}.
\item \texttt{above\_display\_skip\_primitive\_code: 224, 225, 226, 1203, 1206}.
\item \texttt{\textbackslash{}abovewithdelims\_primitive: 1178}.
\item \texttt{abs: 66, 180, 211, 218, 219, 418, 422, 448, 501, 610, 663, 675, 718, 737, 757, 758, 759, 831, 836, 849, 859, 944, 948, 1029, 1030, 1056, 1076, 1078, 1080, 1083, 1093, 1110, 1120, 1127, 1149, 1243, 1244, 1377}.
\item \texttt{absorbing: 305, 306, 339, 473}.
\item \texttt{accent\_kernel: 155, 191, 1125}.
\item \texttt{accent\_nodd: 687, 690, 696, 698, 733, 761, 1165, 1186}.
\item \texttt{accent\_nodd\_size: 687, 698, 761, 1165}.
\item \texttt{act\_width: 866, 867, 868, 869, 871}.
\item \texttt{action\_procedure: 1029}.
\item \texttt{active: 162, 818, 819, 829, 843, 854, 860, 861, 863, 864, 865, 873, 874, 875}.
\item \texttt{active\_base: 220, 222, 252, 253, 255, 262, 263, 333, 442, 306, 1152, 1257, 1289, 1315, 1317}.
\item \texttt{active\_char: 207, 344, 506}.
\item \texttt{active\_height: 970, 975, 976}.
\item \texttt{active\_node\_size: 819, 845, 860, 864, 865}.
\item \texttt{active\_width: 823, 824, 829, 843, 861, 864, 866, 868, 970}.
\item \texttt{actual\_looseness: 872, 873, 875}.
\item \texttt{add\_delims\_to: 347}.
\item \texttt{add\_glue\_ref: 203, 206, 430, 802, 881, 996, 1100, 1229}.
\item \texttt{add\_token\_ref: 203, 206, 323, 979, 1012, 1016, 1221, 1227, 1357}.
\item \texttt{additional: 644, 645, 657, 672}.
\item \texttt{adj\_demerits: 226, 836, 859}.
\item \texttt{adj\_demerits\_primitive: 228}.
\item \texttt{adj\_demerits\_code: 226, 237, 238}.
\item \texttt{adj\_just: 576}.
\item \texttt{adj\_just\_head: 162, 888, 889, 1076, 1085, 1199, 1203}.
adjust node: 142, 148, 175, 183, 202, 206, 647, 651, 655, 730, 761, 866, 899, 1100.
adjust ptr: 142, 197, 202, 206, 655, 1100.
adjust space factor: 1034, 1038.
adjust fail: 647, 648, 649, 651, 655, 796, 888, 889, 1076, 1085, 1199.
adjusted box group: 269, 1062, 1083, 1085.
adv past: 1362, 1363.
advance: 209, 265, 266, 1210, 1235, 1236, 1238.
\advance primitive: 265.
advance major fail: 914, 917.
after: 147, 866, 1196.
after assignment: 208, 265, 266, 1268.
after assignment primitive: 265.
after group: 208, 265, 266, 1271.
after group primitive: 265.
after math: 1193, 1194.
after token: 1266, 1267, 1368, 1369.
aire: 560, 561, 563, 576.
align error: 1126, 1127.
align group: 209, 768, 774, 791, 800, 1131, 1132.
align head: 162, 770, 777.
align peek: 773, 774, 785, 799, 1048, 1133.
align ptr: 770, 771, 772.
align stack node size: 770, 772.
alignment of rules with characters: 589.
alpha: 560, 571, 572.
alpha token: 438, 440.
alter aux: 1242, 1243.
alter box dimen: 1242, 1247.
alter integer: 1242, 1246.
alter page so far: 1242, 1245.
alter prev graf: 1242, 1244.
Ambiguous...: 1183.
Ambie, Ole: 925.
AmSTeX: 1331.
any node: 1045, 1048, 1057, 1063, 1067, 1073, 1097, 1102, 1104, 1126, 1134, 1210, 1238, 1271, 1274, 1276, 1285, 1290, 1347.
any state plus: 344, 345, 347.
append chernode to s: 908, 911.
append choices: 1171, 1172.
append discretionary: 1116, 1117.
append glue: 1057, 1060, 1078.
append italic correction: 1112, 1113.
append kern: 1057, 1061.
append normal space: 1030.
append penalty: 1102, 1103.
append to name: 519*523.
append to plist: 679, 799, 888, 1076, 1203, 1204, 1205.
arena delimiter: 513, 515, 516*517.
arg: 36*.
arg size: 1380*.
arg: 36*.
Argument of \x has...: 395.
arg: 1380*.
arith error: 104.
assign dimen: 209, 248, 249, 413, 1210, 1224, 1228.
assign font dimen: 209, 265, 266, 413, 1210, 1253.
assign font int: 209, 413, 1210, 1253, 1254, 1255.
assign glue: 209, 226, 227, 413, 782, 1210, 1224, 1228.
assign int: 209, 238, 229, 143, 1210, 1222, 1224, 1228, 1237.
assign mw glue: 209, 216, 227, 413, 1210, 1222, 1224, 1228, 1237.
at: 1258.
\atop primitive: 1178.
atop code: 1178, 1179, 1182.
\atop withdelims primitive: 1178.
attach sign: 448, 449, 455.
auto breaking: 862*863, 866, 868.
aux: 212, 213, 216, 800, 812.
aux field: 212, 213, 218, 775.
aux save: 800, 812, 1206.
avail:  118, 120, 121, 122, 123, 164, 168
AJAV list clobbered...:  168.
axis_height:  700, 706, 736, 746, 747, 749, 762.
b:  464, 465, 470, 498, 529, 560, 879, 970, 994
   1198, 1247, 1288.
b_close:  28, 560, 642.
b_make_name_string:  525, 532.
b_open_in:  27*532.
b_open_out:  27*532.
back_error:  327, 373, 396, 403, 415, 442, 446, 476, 479, 503, 57, 783, 1084, 1161, 1197, 1207, 1212.
back_list:  325, 325, 337, 407, 1288.
backed_up:  307, 311, 312, 314, 323, 324, 325, 1026.
background:  823, 824, 827, 837, 863, 864.
backup/backup:  366.
backup_head:  162, 366, 407.
BAD:  293, 294.
bad:  13, 14, 111, 290, 522, 1249, 1332*.
Bad \patterns:  9011.
Bad \preygraf:  1244.
Bad character code:  434.
Bad delimiter code:  437.
Bad flag...:  170.
Bad link...:  182.
Bad mathchar:  436.
Bad number:  435.
Bad register code:  433.
Bad space factor:  1243.
bad_fran:  1303, 1306, 1308, 1312, 1317, 1327*.
bad_pool:  51*52, 53*.
bad_fmn:  560.
badness:  105, 860, 867, 674, 678, 828, 852, 833, 975, 1007.
\badness primitive:  416.
badness_code:  416, 424.
banner:  2*61, 536, 1299.
base_line:  619, 623, 624, 628.
base_ptr:  84*85, 310, 311, 312, 313, 1131, 1380*.
baseline_skip:  224, 247, 679.
\baselineskip primitive:  226.
baseline_skip_code:  149, 224, 225, 226, 679.
baseunix:  1381*.
batch_mode:  73, 75, 86*, 90, 92, 93, 353, 1262, 1263, 1327*, 1328.
\batchmode primitive:  1262.
be:  540, 541, 543, 545, 560, 565, 566, 570, 576.
bech_label:  540, 573, 576.
bchar:  560, 573, 576, 901, 903, 905, 906, 908, 911, 913, 916, 917, 1032, 1034, 1037, 1038, 1040.
bech_label:  540, 552, 576, 900, 916, 1034, 1040, 1322, 1323.
before:  147, 192, 1196.
begin:  77, 8, 1380*.
begin_box:  1073, 1079, 1084.
begin_file_reading:  78, 87, 328, 483, 537*.
begin_group:  208, 265, 266, 1063.
begin_item:  208.
begin_item:  208.
begin_pseudo:  316, 318, 319.
begin_token_list:  322, 359, 386, 390, 774, 788, 789, 799, 1025, 1030, 1083, 1091, 1139, 1145, 1167, 1371.
Beginning to dump...:  1328.
below_display_short_skip:  224.
\belowdisplayshortskip primitive:  226.
below_display_short_skip_code:  224, 225, 226, 1203.
below_display_short:  224.
\belowdisplayprimitive:  226.
below_display_short_code:  224, 225, 226, 1203, 1206.
best_box:  872, 874, 875, 877*878.
best_height_plus_depth:  971, 974, 1010, 1011.
best_ins_ptr:  951, 1005, 1009, 1018, 1020, 1021.
best_line:  872, 874, 875, 877*890.
best_page_break:  950, 1005, 1013, 1014.
best_pline:  833, 845, 855.
best_place:  833, 845, 855, 970, 974, 980.
best_size:  980, 1005, 1017.
beta:  560, 571, 572.
big_p_spacing1:  701, 751.
big_p_spacing2:  701, 751.
big_p_spacing3:  701, 751.

classification set dependencies: 23* 40.

check sum: 53* 542, 588.

check byte range: 570, 573.

check dimensions: 726, 727, 733, 754.

check existence: 573, 574.

check full save stack: 273, 274, 276, 280.

check interrupt: 96, 324, 343, 753, 911, 1031, 1040.

check mem: 165, 167, 1031, 1390.

class outer validity: 336, 351, 353, 354, 357, 362, 375.

check shrinking: 825, 827, 868.

Chinese characters: 134, 555.

class node: 688, 689, 690, 698, 730.

choose mist: 731.

class: 19, 20, 23* 24, 519, 523, 1222, 1380.

class end: 208, 781.


class q: 212, 213, 812, 1034, 1091, 1200, 1376, 1377.

clean box: 730, 734, 735, 737, 738, 742, 744, 749, 750, 757, 758, 759.


clear io result: 22*.

clear terminal: 31* 330, 530* 1338.

COBBERED: 293.


close: 28.

close files and terminate: 78, 81, 1332* 1333.

}\closein primitive: 1272.

\close node: 682, 690, 696, 698, 728, 761, 762, 1156, 1157.

close node: 1341, 1344, 1346, 1348, 1356, 1357, 1358, 1373, 1374, 1375.

\closeout primitive: 1344.

closed: 480, 481, 483, 485, 486, 501, 1275.

\chr: 737, 743, 745, 746, 756, 757, 758, 759.

\club penalty: 236, 890.

\clubpenalty primitive: 238.

\clubpenalty code: 236, 237, 238.
1351, 1374.  
cur_fam: 236, 1151, 1155, 1165.  
cur_fam_code: 236, 237, 238, 1139, 1145.  
cur_file: 304, 329, 362, 373* 38.  
cur_font: 239, 232, 558, 559, 577, 1032, 1034, 1042, 1044, 1117, 1123, 1124, 1146.  
cur_g: 619, 625, 629, 634.  
cur_glia: 619, 625, 629, 634.  
cur_group: 270, 271, 272, 274, 281, 282, 800, 1062, 1063, 1064, 1065, 1067, 1068, 1069, 1130, 1131, 1140, 1142, 1191, 1192, 1193, 1194, 1200.  
cur_h: 616, 617, 618, 619, 620, 622, 623, 626, 627, 628, 629, 632, 637.  
cur_head: 770, 771, 772, 786, 799.  
cur_height: 970, 972, 973, 974, 975, 976.  
cur_i: 722, 723, 724, 738, 741, 749, 752, 753, 755.  
cur_if: 336, 489, 490, 495, 496, 1335.  
cur_indent: 877* 889.  
cur_input: 35, 36, 87, 301, 302, 311, 321, 322, 324, 1131.  
cur_J: 907, 908, 909, 910, 911, 1032, 1034, 1035, 1036, 1037, 1039, 1040.  
cur_lang: 891, 892, 923, 924, 930, 934, 939, 944, 963, 1091, 1200, 1362.  
cur_level: 270, 271, 272, 274, 277, 278, 280, 281, 1304, 1335.  
cur_line: 877* 889, 890.  
cur_list: 213, 216, 217, 218, 422, 1244.  
cur_lop: 770, 771, 772, 777, 783, 792, 793, 794.  
cur_mark: 296, 382, 386, 1335.  
cur_nlist: 719, 720, 726, 754, 1194, 1196, 1199.  
cur_mu: 703, 719, 730, 732, 766.  
cur_name: 512, 517, 529, 530* 537* 1257, 1258, 1260, 1351, 1374.  
cur_order: 366, 439, 447, 448, 454, 462.  
cur_q: 907, 908, 910, 911, 1034, 1035, 1036, 1037, 1040.  
cur_r: 907, 908, 909, 910, 911, 1032, 1034, 1037, 1038, 1039, 1040.  
curRh: 906, 908, 909, 910.
\texttt{\textbackslash day} primitive: 238
\texttt{\textbackslash day\_code}: 236, 237, 238.
d\_d: 241*
d\_d: 458.
d\_de: 1380*
d\_deac\_tiv\_a\_te\_cycles: 829, 831, 834.
d\_deac\_tiv\_a\_te\_cycles\: primitive: 416.
d\_debug: 7\*9*78, 84*93, 114, 165, 166, 167, 172, 1031, 1338*.
d\_debug\_#: 1338*.
d\_debug\_help: 78, 84*93, 1338*.
d\_debug\_ing: 7\*84*96, 114, 165, 182, 1031, 1338*.
d\_decent\_fit: 817, 834, 852, 853, 864.
d\_decode\_data: 241*.
d\_decode\_time: 241*.
d\_deer: 16, 42, 44, 64, 71, 86*88, 89, 90, 92, 102, 120, 121, 123, 175, 177, 200, 201, 205, 217, 245, 260, 281, 282, 311, 322, 324, 325, 329, 331, 347, 356, 357, 360*.
d\_def: 209, 1208, 1209, 1210, 1213, 1218.
d\_def\_code: 209, 413, 1210, 1230, 1231, 1232.
d\_def\_family: 209, 413, 577, 1210, 1230, 1231, 1234.
d\_def\_font: 209, 265, 266, 413, 577, 1210, 1230, 1231, 1234.
d\_default\_code: 683, 697, 743, 1182.
d\_default\_hyphen\_char: 236, 576.
\texttt{\textbackslash default\_hyphen\_char} primitive: 238.
d\_default\_hyphen\_char\_code: 236, 237, 238.
d\_default\_rule: 463.
d\_default\_rule\_thickness: 683, 701, 734, 735, 737, 743, 745, 759.
d\_default\_skew\_char: 236, 576.
\texttt{\textbackslash default\_skew\_char} primitive: 238.
\texttt{\textbackslash default\_skew\_char\_code}: 236, 237, 238.
def\_ac\_t\_ion: 597*.
d\_define: 1214, 1217, 1218, 1221, 1222, 1224, 1225, 1226, 1227, 1228, 1232, 1234, 1236, 1248, 1257.
def\_in\_g\_i\_n\_g: 305, 306, 339, 473, 482.
d\_del\_code: 236, 240, 1160.
\texttt{\textbackslash del\_code} primitive: 1320.
d\_delete\_glue\_ref: 201, 202, 275, 451, 465, 578, 732, 802, 816*826, 881, 976, 996, 1004, 1017, 1022, 1100, 1229, 1236, 1239, 1335.
d\_delete\_last: 1104, 1105.
d\_delete\_q: 726, 760, 763.
d\_delete\_token\_ref: 200, 202, 275, 324, 977, 979, 1012, 1016, 1335, 1338.
d\_de\_lets\_a\_l\_low\_ed: 76, 77, 84*85, 98, 336, 346.
d\_del\_im\_num: 207, 265, 266, 1046, 1151, 1154, 1160.
d\_del\_im\_t\_ed\_c\_od\_e: 1178, 1179, 1182, 1183.
d\_del\_im\_t\_er: 687, 696, 762, 1191.
\texttt{\textbackslash del\_im\_t\_er} primitive: 236, 762.
\texttt{\textbackslash del\_im\_t\_er\_f\_act\_or\_i\_t\_e\_r\_i\_v\_e}: 238.
\texttt{\textbackslash del\_im\_t\_er\_f\_act\_or\_i\_t\_e\_r\_i\_v\_e\_c\_od\_e}: 236, 237, 238.
\texttt{\textbackslash del\_im\_t\_er\_s\_h\_o\_r\_t\_f\_a\_l\_l}: 247, 762.
\texttt{\textbackslash del\_im\_t\_er\_s\_h\_o\_r\_t\_f\_a\_l\_l\_c\_od\_e}: 247, 248.
d\_del\_im\_1: 700, 748.
d\_del\_im\_2: 700, 748.
\textit{depth_base}: 550, 552, 554, 566, 571, 1322, 1323.

\textit{depth_index}: 543, 554

\textit{depth_offset}: 135, 416, 769, 1247.

\textit{depth_threshold}: 181, 182, 198, 233, 236, 692, 1339^*

\textit{dig}: 54, 64, 65, 67, 102, 452.

\textit{digit_sensed}: 960, 961, 962.

\textit{dimen}: 247, 427, 1008, 1010.

\textit{dimen/primitive}: 411.

\textit{dimen_base}: 220, 226, 247, 248, 249, 250, 251, 252, 1070, 1145.

\textit{dimen_def_code}: 1222, 1223, 1224.

\textit{dimen_pdir}: 247.

\textit{dimen_pars}: 247.


Dimension too large: 460.

dirty Pascal: 3, 114, 172, 182, 186, 285, 811, 1331, 1382^*

\textit{disc_break}: 877^*880, 881, 882, 890.

\textit{disc_group}: 209, 1117, 1118, 1119.


\textit{disc_width}: 839, 840, 869, 870.

\textit{discretionary}: 308, 1090, 1114, 1115, 1116.

\textit{Discretionary list is too long}: 1120.

\textit{discretionary primitive}: 1114.

Display math...with $\$: 1197.

\textit{displayindent}: 247, 800, 1138, 1145, 1199.

\textit{displayindent primitive}: 248.

\textit{displayindent code}: 247, 248, 1145.

\textit{displaylimits primitive}: 1156.

\textit{displaylist}: 689, 699, 699, 731, 1174.

\textit{displaystyle}: 689, 694, 731, 1169, 1199.

\textit{displaystyle primitive}: 1169.

\textit{displaywidow_penalty}: 236, 1145.

\textit{displaywidow_penalty primitive}: 238.

\textit{displaywidow_penalty code}: 236, 237, 238.

\textit{display_width}: 247, 1138, 1145, 1199.

\textit{displaywidth primitive}: 248.

\textit{displaywidth code}: 247, 248, 1145.

\textit{div}: 100, 627, 636.

\textit{divide}: 209, 265, 266, 1210, 1235, 1236.

\textit{divide primitive}: 265.

\textit{do all six}: 823, 829, 832, 837, 843, 844, 860, 861, 864, 970, 987.

\textit{do/assignments}: 800, 1123, 1206, 1270.

\textit{do/extend}: 1130, 1131.

\textit{do_extension}: 1347, 1348, 1375.

\textit{do/nothing}: 16, 27*33*34*57, 58, 84*175, 202, 275, 344, 357, 538, 569, 699, 611, 612, 622, 631, 651, 669, 692, 728, 733, 761, 837, 866, 899, 1045, 1236, 1359, 1360, 1373.

\textit{do_register_command}: 1235, 1236.

\textit{doing/leaders}: 592, 593, 628, 637, 1374.


\textit{done/with/node}: 726, 727, 728, 733, 754.

\textit{done/with/node}: 726, 727, 730, 731, 754.


\textit{done3}: 15, 815, 897, 898.

\textit{done4}: 15, 815, 899.

\textit{done5}: 15, 815, 866, 869.

\textit{done6}: 15.

\textit{dont/expand}: 210, 258, 357, 369.

\textit{Double subsuptext}: 1177.

\textit{Double superscript}: 1177.

\textit{double/\textbf{hyphen/\textit{demerits}} primitive}: 236, 859.

\textit{double/\textbf{hyphen/\textit{demerits/\textbf{code}}}}: 236, 237, 238.

\textit{Doubly free location... 169.

\textit{down_prt}: 695, 696, 697, 615.

\textit{down\_date/width}: 860.

\textit{down1}: 585, 586, 607, 613, 614, 616.

\textit{down2}: 585, 594, 610.

\textit{down3}: 585, 610.

\textit{down4}: 585, 610.

\textit{\textbf{dp/primitive}}: 416.

\textit{dry/rot}: 95.

\textit{dump...only by \textbf{INITEX}}: 1335.

\textit{dump primitive}: 1052.

\textit{dump\_four/\textbf{ASCII}}: 1309.

\textit{dump\_hh}: 1305, 1318, 1324.

\textit{dump\_int}: 1305, 1307, 1309, 1311, 1313, 1315, 1316, 1318, 1320, 1322, 1324, 1326.
dump_qqqq: 1305, 1309, 1322.
dump qed: 1305, 1311, 1315, 1316, 1320.
Duplicate pattern: 963.
dvi buf: 594, 595, 597*598, 607, 613, 614.
dvi buf size: 11*14, 594, 595, 596, 598, 599, 607, 613, 614, 642.
dvi f: 616, 617, 620, 621.
dvi file: 532, 539, 595, 597*642.
DVI files: 583.
dvi font def: 600, 621, 643.
dvi four: 600, 602, 610, 617, 624, 633, 640, 642, 1368.
dvi gone: 594, 595, 596, 598, 612.
dvi index: 594, 595, 597*
dvi limit: 594, 595, 596, 598, 599.
dvi pop: 601, 619, 629.
dvi swap: 598.
dvi v: 616, 617, 619, 623, 628, 629, 632, 637.
dyn used: 117, 120, 121, 122, 123, 164, 639, 1311, 1312.
e: 277, 279, 518, 519*530*1198, 1211.
easy line: 819, 835, 847, 848, 850.
ec: 540, 541, 543, 545, 550, 555, 566, 570, 576.
ed: 1380*
edef primitive: 1208.
edge: 619, 623, 626, 629, 635.
edit: 1380*
edit file: 1380*
Editor: 79*80*537*1380*
editor: 1380*
editor arg: 1380*
editor length: 1380*
eight bits: 25*64, 112*297, 549, 560, 581, 582, 595, 607, 649, 706, 709, 712, 977, 992, 993, 1079, 1247, 1288.
eject penalty: 157, 829, 831, 851, 859, 873, 970, 972, 974, 1005, 1010, 1011.
else: 10*
\else primitive: 491.
else code: 489, 491, 496.
em: 455.
Emergency stop: 93.
emergency stretch: 247, 828, 863.
\emergencystretch primitive: 248.
emergency stretch code: 247, 248.
empty line at end of file: 486, 538.
empty field: 684, 685, 686, 742, 1163, 1165, 1181.
empty flag: 124, 126, 130, 150, 164, 1312.
end: 7*8, 10*1380*
End of file on the terminal: 71.
(\end occurred...): 1335.
\end primitive: 1052.
end cs name: 208, 265, 266, 372, 1134.
\endcsname primitive: 265.
end file reading: 329, 330, 360*362, 483, 537*1335.
end graf: 1026, 1085, 1094, 1096, 1100, 1131, 1133, 1168.
end group: 208, 265, 266, 1063.
\endgroup primitive: 265.
\endinput primitive: 376.
end line char: 87, 236, 240, 303, 318, 332, 360*362, 483, 534, 538, 1337.
\endlinechar primitive: 228.
end line char code: 236, 237, 238.
end line char inactive: 360*362, 483, 538, 1337.
end match: 207, 289, 291, 294, 391, 392, 394.
end match token: 289, 389, 391, 392, 393, 394, 474, 476, 482.
end name: 512, 517, 526, 531.
end of TEX: 6, 81, 1332*.
end span: 162, 768, 779, 793, 797, 801, 803.
end template: 210, 366, 375, 380, 780, 1295.
end template token: 780, 784, 790.
end token list: 324, 325, 357, 390, 1026, 1335, 1371.
end write: 222, 1369, 1371.
\endwrite: 1369.
\endwrite token: 1371, 1372.
endcases: 10*.
endtemplate: 780.
endw: 207, 298, 375, 380, 768, 780, 782, 791, 1046, 1130, 1131.
ensure dvi open: 532, 617.
ensure pbox: 203, 1009, 1018.
\eof: 26, 31*52, 564, 1338*1339*.
font_name_code: 468, 469, 471, 472.
font_params: 549, 552, 576, 578, 579, 580, 1195, 1322, 1323.
font_size: 472, 549, 552, 568, 602, 1260, 1361, 1322, 1323.
font_used: 549, 551, 621, 643.
FONTx: 1257.
for accent: 191.
Forbidden control sequence...: 338.
free_eof: 331, 361, 362, 378.
format_area_length: 520, 524*.
format_default_length: 520, 522, 523, 524*.
format_ext_length: 520, 523, 524*.
format_extension: 520, 599, 1328.
format_ident: 35, 61, 536, 1299, 1300, 1301, 1326, 1327*1328, 1337.
found2: 15, 895, 902, 1302, 1315.
found2: 15, 895, 903, 1302, 1316.
four_choices: 113.
fpexecvp: 1380*.
fp_signal: 1381*.
FPC Pascal: 9*10*27*34*36*109*241*523*1323*1380*1381*1382*.
FPC-bufser: 597*.
FPC-EOF: 575*1327*.
fp_assign: 27*.
fp_blockwrite: 597*.
fp_decode_date: 241*.
fp_decode_time: 241*.
fp_flush: 34*.
fp_fpexecvp: 1380*.
fp_fpgetserrno: 1381*.
fp_fpgets_signal: 1381*.
fp_halt: 1322*.
fp-io_result: 27*.
fp_lengeth: 36*.
fp_now: 241*.
fp_params_count: 36*.
fp_params_str: 36*.
fp_pchar: 1380*.
fp_SIGINT: 1381*.
fp_signal_handler: 1381*.
fp_single: 109*.
fp_string: 36*.
fraction_nod: 683, 687, 690, 698, 733, 761, 1178, 1181.
fraction_nod_size: 683, 698, 761, 1181.
fraction_rule: 704, 705, 733, 747.
freeze: 211, 212, 213, 217, 400, 452, 772, 915, 1036, 1226, 1288.
freenode: 130, 201, 202, 275, 496, 615, 655, 698, 715, 721, 727, 751, 753, 756, 760, 772, 803, 860, 861, 865, 903, 910, 977, 1019, 1021, 1022, 1037, 1100, 1110, 1169, 1187, 1210, 1335, 1358.
freespec: 987, 1001, 1008.
frozen_control_sequence: 222, 258, 1215, 1314, 1318, 1319.
frozen: 222, 339, 780, 1132.
frozen_font: 222, 258, 369.
frozen_font_group: 222, 265, 1065.
frozen_font_template: 222, 375, 780.
frozen_font_width: 222, 375, 380, 780.
frozen_font_height: 222, 336, 491.
frozen_count: 222, 553.
frozen: 222, 1215, 1216.
frozen: 222, 265, 379.
frozen_right: 222, 1065, 1188.
Fuchs, David Raymond: 2*583, 591.
\futurelet primitive: 1219.
g: 47, 182, 260, 592, 649, 668, 706, 716.
g_order: 619, 625, 629, 634.
gsign: 619, 625, 629, 634.
garbage: 162, 467, 470, 960, 1183, 1192, 1279.
\gdef primitive: 1308.
gdef: 279, 782, 1077, 1214.
seq_word: 279, 288, 1013, 1214.
get: 9*26, 29, 31, 33, 3485, 538, 564, 1306.
gt_find: 125, 131, 136, 139, 144, 145, 147, 151, 152, 153, 156, 158, 206, 495, 607, 649, 668, 686, 689, 716, 772, 798, 843, 844, 845, 864, 914, 1009, 1100, 1101, 1163, 1165, 1181, 1248, 1249, 1349, 1357.
gt_preamble: 782, 783, 784.
get_r_token: 1215, 1218, 1221, 1224, 1225, 1257.
get_strings_started: 47, 51, 1332*
get_r_token_or_active_char: 506.
give_err_char: 78, 89, 90, 1284.
global: 1214, 1218, 1241.
global definitions: 221, 279, 283.
\global primitive: 1208.
\global def: 226, 782, 1214, 1218.
\globaldefs primitive: 238.
\globaldefs code: 230, 237, 238.
glue base: 220, 222, 224, 226, 227, 228, 229, 252, 782.
glue_offset: 135, 139, 186.
glue par: 224, 766.
glue params: 224.
glue ratio: 109*110, 113, 135, 186.
glue ref: 210, 228, 275, 782, 1228, 1236.
glue ref count: 150, 151, 152, 153, 154, 164, 201, 203, 228, 766, 1043, 1060.
glue shrink: 135, 185, 796, 799, 801, 810, 811.
glue spec size: 150, 151, 162, 164, 201, 716.
glue stretch: 159, 185, 796, 799, 801, 810, 811.
glue temp: 619, 625, 629, 634.
goal height: 986, 987.
goto: 35, 81.
gt: 110, 113, 114, 135.
gubed: 7*
Guibas, Leonidas Ioannis: 2*
g1: 1198, 1203.
g2: 1198, 1203, 1205.
h_offset: 247, 617, 641.
h_offset primitive: 248.
h_offset code: 247, 248.
ha: 892, 896, 900, 903, 912, 1381*.
half: 100, 706, 736, 737, 738, 745, 746, 749, 750, 1202.
half buf: 594, 595, 596, 598, 599.
half error line: 11*14, 311, 315, 316, 317.
half align: 208, 265, 266, 1004, 1130.
\halfalign primitive: 265.
half: 1380*.
handle_right_brace: 1067, 1068.
hang after: 236, 240, 847, 849, 1070, 1149.
\hangafter primitive: 238.
hang after code: 236, 237, 238, 1070.
\hangindent primitive: 248.
\hangindent code: 247, 248, 1070.
\hspace: 234, 256, 257, 259, 260, 1318, 1319.
\hskip: 220, 222, 226, 257, 259, 262, 263, 1257, 1314, 1318, 1319.
\hspace: 473, 476.
\hskip: 256, 260.
\hskip: 12, 14, 259, 261, 1307, 1308.
\hsize: 12, 14, 222, 260, 261, 1334.
\hspace used: 256, 258, 260, 1318, 1319.
\hspace: 892, 897, 898, 900, 903.
hbadness: 236, 660, 666, 667.
\hbadness primitive: 238.
\indent primitive: 1088

indent \hmode: 1092, 1093

indented: 1091.

index: 300, 302, 303, 304, 307, 328, 329, 331.

index_field: 300, 302, 1131.

inf: 447, 448, 453.


Infinite glue shrinkage: 826, 976, 1004, 1009.

infinity: 445.


init: 8, 47, 50, 131, 264, 891, 942, 943, 947, 950, 1252, 1302, 1325, 1332*1335, 1336.

init_align: 773, 774, 1130.

init_col: 773, 785, 788, 791.

init_cur_lang: 816*891, 892.

init_\hif: 816*891, 892.

init_fnt: 900, 903, 905, 908.

init_ig: 900, 903, 905, 908.

init_list: 900, 903, 905, 908.

init_math: 1137, 1138.

init_pool_ptr: 39, 42, 1310, 1332*1334.

init_prim: 1332*1336.

init_\ryf: 816*891, 892.

init_row: 773, 785, 786.

init_son: 773, 786, 787, 791.

init_str_ptr: 39, 43, 517, 1310, 1332*1334.

init_terminal: 37*331.

init_tric: 891, 966, 1324.

initex: 8, 117*12, 47, 50, 116, 1299, 1331.

initialize: 4*1332*1337.

inner loop: 31*112*120, 121, 122, 123, 125, 127, 128, 130, 202, 324, 325, 341, 342, 343, 357, 365, 380, 399, 407, 554, 597*.


inner_noad: 652, 683, 690, 696, 698, 733, 761, 764, 1156, 1157, 1191.


input primitive: 376.

input_command_in: 36*37*.

input_file: 304.

inputlineno primitive: 416.

input_line_no_code: 416, 417, 424.
input_fn: 30, 31*36*37*58, 71, 362, 485, 486, 538.
inout_ptr: 301, 311, 312, 321, 322, 330, 331, 360*534, 1131, 1335.
inout_stack: 84*55, 301, 311, 321, 322, 534, 1131, 1380.
ins_disc: 1032, 1033, 1035.
ins_error: 327, 336, 395, 1047, 1127, 1132, 1215.
ins_list: 323, 339, 467, 470, 1064, 1371.
ins_node: 140, 148, 175, 183, 202, 206, 647, 651, 730, 761, 866, 899, 968, 973, 981, 986, 1000, 1014, 1100.
ins_node_size: 140, 202, 206, 1022, 1100.
inout_ptr: 140, 188, 202, 206, 1010, 1020, 1021, 1100.
ins_the_foks: 366, 367, 467.
insert: 208, 265, 266, 1097.
insert>: 87.
\insert primitive: 265
insert_dollar_sign: 1045, 1047.
insert_group: 269, 1068, 1099, 1100.
insert_penalties: 419, 982, 990, 1005, 1008, 1010, 1014, 1022, 1026, 1242, 1246.
\insertpenalties primitive: 416
insert_token: 268, 280, 282.
inserted: 307, 314, 323, 324, 327, 379, 1095.
inserting: 981, 1009.
Insertions can only... 993.
inserts_only: 980, 987, 1008.
int_par: 236.
int_pars: 236.
inter_line_penalty: 236, 890.
\interlinepenalty primitive: 238
inter_line_penalty-code: 236, 237, 238.
interaction: 71, 72, 73, 74, 75, 82, 83, 84* 86*90, 92, 93, 98, 360*363, 484, 530* 1265* 1283, 1293, 1294, 1297, 1326, 1327* 1328, 1335.
interrupt: 4*96, 97, 98, 1031, 1382*
Interruption: 98.
terroot alignment preambles...: 324 * 782, 789, 791, 1131.
Invalid code: 1232.
invalid_char: 207, 232, 344.
invalid_code: 22, 24, 36*232.
is_empty: 124, 127, 169, 170.
is_hex: 352, 355.
is_running: 138, 176, 624, 633, 806.
ISO: 9*.
issue_message: 1276, 1279.
italcorr: 208, 265, 266, 11111, 1112.
italic correction: 543.
italic_base: 550, 552, 554, 566, 571, 1322, 1323.
italic_index: 543.
its_all_over: 1045, 1054, 1335.
iv: 1380*.
j: 45, 46, 59, 60, 69, 70, 259, 264, 315, 366, 519*523*524*638, 893, 901, 906, 934 966, 1211, 1302, 1303, 1348, 1370, 1373.
Jensen, Kathleen: 10*.
max_strings: 11*38, 43, 111, 517, 525, 1310, 1334.
\max_r: 592, 593, 641, 642
\meaning primitive: 468
\meaning_code: 468, 469, 471, 472.
\medmathskip: 224
\medsmallskip primitive: 226
\medsmallskip code: 224, 225, 226, 766.
\mem: 11*12, 14, 111, 116, 125, 126, 162, 164, 1307, 1308, 1311, 1312.
\mem: 11*12, 14, 110, 111, 116, 120, 124, 125, 165, 166.
\mem: 11*12, 14, 111, 116, 162, 164, 1249, 1307, 1308, 1312.
Memory usage... 639.
\memory: 110, 113, 114, 116, 182, 212, 218, 221, 253, 268, 271, 275, 548, 549, 800, 1305.
\message: 208, 1276, 1277, 1278.
\message: 1277.
\message: 589.
\mid: 546
\mid: 87, 203, 328, 344, 347, 352, 353, 354.
\min: 11*110, 111, 113, 115, 230, 1027, 1323, 1325.
\min: 208, 413, 440, 448, 455, 461.
\minimal: 833, 834, 836, 845, 855.
\minimum: 833, 834, 835, 836, 854, 855.

minor_tail: 912 915, 916.
minus: 462
Misplaced & : 1128.
Misplaced \cr : 1128.
Misplaced \noalign: 1129.
Misplaced \omit: 1129.
Misplaced \span: 1128.
Missing = inserted: 303.
Missing \$ inserted: 1047, 1065.
Missing \cr inserted: 1132.
Missing \endcsname... 373.
Missing \endgroup inserted: 1065.
Missing \right. inserted: 1065.
Missing \{ inserted: 403, 475, 1127.
Missing \} inserted: 1065, 1127.
Missing ‘to’ inserted: 1082.
Missing ‘to’... 1225.
Missing \$\$ inserted: 1207.
Missing character: 581.
Missing control... 1215.
Missing delimiter... 1161.
Missing font identifier: 577.
Missing number... 415, 446.
mkern: 208, 1046, 1057, 1058, 1059.
\mkern primitive: 1058.
\ml_field: 212, 213, 218.
\mlist: 726, 760.
\mlist_penalties: 719, 720, 726, 754, 1194, 1196, 1199.
\mlist_to_hlist: 693, 719, 720, 725, 726, 734, 754, 760, 1194, 1196, 1199.
\mm: 241*
\mm: 458.
\mmode: 211, 212, 213, 218, 501, 718, 775, 776, 800, 812, 1030, 1045, 1046, 1048, 1056, 1057, 1073, 1080, 1092, 1097, 1109, 1110, 1112, 1116, 1120, 1130, 1136, 1140, 1145, 1150, 1154, 1158, 1162, 1164, 1167, 1171, 1175, 1180, 1190, 1193, 1194.
\mode: 211, 212, 213, 215, 216, 299, 418, 422, 424, 501, 718, 775, 776, 785, 786, 787, 796, 799, 804, 807, 808, 809, 812, 1025, 1029, 1030, 1034, 1035, 1049, 1051, 1056, 1076, 1078, 1080, 1083, 1086, 1091, 1093, 1094, 1095, 1096, 1099, 1103, 1105, 1110, 1117, 1119, 1120, 1136, 1138, 1145, 1167, 1194, 1196, 1200, 1243, 1370, 1371, 1377.
MODE: 9*
\mode_field: 212, 213, 218, 422, 800, 1244.
\mode_line: 212, 213, 215, 216, 304, 804, 815, 1025.
new_spec: 151, 154, 430, 462, 826, 976, 1004, 1042, 1043, 1239, 1240.
new_string: 54, 57, 58, 465, 470, 617, 1257, 1279, 1328, 1368, 1380*
new_style: 688, 1171.
new_trie_op: 943, 944, 945, 965.
new_whatisit: 1349, 1350, 1354, 1376, 1377.
new_write_whatisit: 1350, 1351, 1352, 1353.
next: 256, 257, 259, 260.
next_break: 877*878.
next_char: 545, 741, 753, 909, 1039.
next_p: 619, 622, 626, 629, 630, 631, 633, 635.
nh: 540, 541, 560, 565, 566, 569.
ni: 540, 541, 560, 565, 566, 569.
nil: 16.
nk: 540, 541, 560, 565, 566, 573.
nl: 50, 540, 541, 545, 560, 565, 566, 569, 573, 576.
nn: 311, 312.
no_pages_of_output: 642.
no_align: 208, 265, 266, 785, 1126.
\noalign primitive: 265.
no_align_error: 1126, 1129.
no_align_group: 269, 768, 785, 1133.
no_boundary: 208, 265, 266, 1030, 1038, 1045, 1090.
\nobreak primitive: 265.
no_breakYet: 829, 836, 837.
\noexpand primitive: 265.
no_expand_flag: 358, 506.
\noindent primitive: 1088.
no_limits: 682, 1156, 1157.
\no_limits primitive: 1156.
no_new_control_sequence: 236, 257, 259, 264, 365, 374, 1336.
no_print: 54, 57, 58, 75, 98.
no_shrink_error Yet: 825, 826, 827.
no_tag: 544, 505.
nod席_size: 681, 686, 698, 753, 761, 1186, 1187.
nod席_list_display: 180, 184, 188, 190, 195, 197.
nod席_what stays_active: 830, 851, 854.
nod席_size: 124, 126, 128, 130, 164, 169, 1311, 1312.
nom: 560, 561, 563, 576.
non_address: 549, 552, 576, 909, 916, 1034.
non_char: 549, 552, 576, 897, 898, 901, 908, 909, 910, 911, 915, 916, 917, 1032, 1034, 1035, 1038, 1039, 1040, 1323.
non_discardable: 148, 879*.
non_math: 1046, 1063, 1144.
non_prunable_p: 816*862*876*877*879*.
non_script: 208, 265, 266, 1046, 1171.
\nonscript primitive: 265, 732.
none_seen: 611, 612.
NONEXISTENT: 262.
Nonletter: 962.
nonstop_mode: 73, 86*, 860*, 863, 484, 1262, 1263.
\nonstopmode primitive: 1262.
nop: 583, 585, 586, 588, 590.
norm_min: 1001, 1200, 1376, 1377.
normal_paragraph: 774, 785, 787, 1025, 1070, 1083, 1094, 1096, 1099, 1167.
normalize-selector: 78, 92, 93, 94, 95, 863.
Not a letter: 937.
not_found: 15, 45, 46, 448, 455, 560, 570, 607, 611, 612, 895, 930, 931, 934, 941, 953, 955, 970, 973, 1138, 1146, 1365.
notexpanded: 238.
now: 241*
null delimiter: 240, 1065.
null_character: 555, 556, 722, 723.
null_code: 22, 232.
nullcs: 222, 262, 263, 354, 374, 1257.
null_delimiter: 684, 685, 1181.
null_delimiter_space: 247, 706.
nulldelimiterspace primitive: 248
null_font: 232, 552, 553, 560, 577, 617, 663,
706, 707, 722, 864, 1257, 1320, 1321, 1339*.
nullfont primitive: 553
null_list: 14, 162, 380, 780.
um: 450, 458, 583, 587, 590.
um_style: 702, 744.
Number too big: 445.
number primitive: 468
number_code: 468, 490, 470, 471, 472.
numerator: 683, 690, 697, 698, 744, 1181, 1185.
num1: 700, 744.
num2: 700, 744.
num3: 700, 744.
num: 540, 541, 560, 565, 566, 569.
num_plus_y: 103, 455, 716, 1240.
offset_token: 438, 444.
odd: 62, 100, 193, 504, 758, 898, 902, 908, 909, 913, 914, 1211, 1218.
aff_save: 1063, 1064, 1094, 1095, 1130, 1131,
1140, 1192, 1193.
OK: 1298.
OK so far: 440, 445.
OK_to_interrupt: 88, 96, 97, 98, 327, 1031.
old_j: 829, 835, 850.
old_mode: 1370, 1371.
old_rover: 131
old_setting: 245, 246, 311, 312, 465, 470,
534, 617, 638, 1257, 1279, 1368, 1370.
omit: 268, 269, 266, 788, 789, 1126.
omit primitive: 265,
omit_error: 1126, 1129.
omit_template: 162, 789, 790.
Only one # is allowed...: 784.
open_byte: 545, 557, 741, 753, 909, 911, 1040.
open_nod: 682, 690, 696, 698, 728, 733,
749, 761, 1156, 1157, 1159.
on_start: 920, 921, 924, 945, 1325.
on_area: 1341, 1351, 1356, 1374.
on_ext: 1341, 1351, 1356, 1374.
on_font_file: 524*, 1337.
onin primitive: 1272
open_log_file: 78, 92, 360*, 471, 532, 534,
535, 537* 1257, 1335.
open_name: 1341, 1351, 1356, 1374.
open_nod: 682, 690, 696, 698, 728, 733,
761, 762, 1156, 1157.
on_node: 1341, 1344, 1346, 1348, 1356,
1357, 1358, 1373.
on_node_size: 1341, 1351, 1357, 1358.
on_or_close_in: 1274, 1275.
onout primitive: 1344
open_paren: 304, 331, 362, 537* 1335.
or primitive: 491
or_code: 480, 491, 492, 500, 509.
ord: 20.
onord: 681, 682, 686, 687, 690, 696, 698,
728, 729, 733, 752, 753, 761, 764, 765,
1075, 1156, 1157, 1186.
order: 177
oriental characters: 134, 585.
other_A_token: 445
other_char: 207, 232, 289, 291, 294, 298,
347, 445, 464, 526, 935, 961, 1030, 1038,
1090, 1124, 1151, 1154, 1160.
onother_token: 289, 405, 438, 441, 445, 464,
503, 1065, 1221.
othercases: 10*
others: 10*
Ouch...clobbered: 1332*
on_parm: 207, 289, 291, 294, 357.
\parindent primitive: 248
\parindent code: 247, 248.
\parloc: 333, 334, 351, 1313, 1314.
\parspace primitive: 265.
\parshape loc: 230, 232, 233, 1070, 1248.
\parshape pbr: 230, 232, 233, 423, 814, 847,
848, 850, 889, 1070, 1149, 1249.
\parskip: 224, 1091.
\parskip primitive: 226.
\parskip code: 224, 225, 226, 1091.
\parshape: 333, 334, 339, 392, 395, 399,
1095, 1314.
Paragraph ended before...: 396.
\parspace: 542, 547, 558.
\parshape base: 550, 552, 558, 566, 574, 575* 576, 578, 550, 700, 701, 1042, 1322, 1323.
\parshape end: 558.
\parshape pbr: 305, 323, 324, 331, 390.
\parshape size: 11*305, 390, 1334.
\parshape stack: 307, 308, 324, 339, 388,
389, 390.
\parshape start: 307, 323, 324, 359.
\parameter: 307, 314, 359.
\parameters for symbols: 700, 701.
\parameters...consecutively: 476.
Pascal-H: 3, 4*9*10*27*28, 33*
Pascal: 1, 10*698, 764.
\pass_number: 821, 845, 864.
\pass_text: 306, 494, 500, 509, 510.
\passe: 821, 845, 846, 864, 865.
\passe_node_size: 821, 845, 865.
Patterns can be...: 1252.
\patterns primitive: 1250.
\pause_for_instructions: 96, 98.
\pausing: 236, 363.
\pausing primitive: 238.
\pausing code: 236, 237, 238.
\pc: 458.
\pch: 1380*.
\pen: 726, 761, 767, 877*890.
\penalties: 1102.
\penalties: 726, 767.
\penalty: 157, 158, 194, 424, 816*866, 973,
996, 1000, 1010, 1011, 1013.
\penalty primitive: 265.
\penalty node: 157, 158, 183, 202, 206, 424,
730, 761, 767, 816*817, 837, 856, 866,
879*899, 996, 973, 996, 1000, 1010,
1011, 1013, 1107.
\pg_field: 212, 213, 218, 219, 422, 1244.
\pi: 829, 831, 851, 856, 859, 970, 972, 973,
974, 994, 1000, 1005, 1006.
\plain: 521*524*1331.
Plass, Michael Frederick: 2*813.
Please type...: 360*530*.
Please use \mathaccent...: 1166.
\PLtoTF: 561.
\plus: 462.
\point_token: 438, 440, 448, 452.
\pointer: 115, 116, 118, 120, 123, 124, 125,
130, 131, 136, 139, 144, 145, 147, 151,
152, 153, 154, 156, 158, 165, 167, 172,
198, 200, 201, 202, 204, 212, 218, 252,
256, 259, 263, 275, 276, 277, 278, 279,
281, 284, 295, 297, 305, 306, 308, 323,
450, 461, 463, 464, 465, 473, 482, 489,
497, 498, 549, 560, 582, 592, 605, 607,
615, 619, 629, 638, 647, 649, 668, 679,
686, 688, 691, 692, 704, 705, 706,
709, 711, 715, 716, 717, 719, 720, 722,
726, 734, 735, 736, 737, 738, 743, 749,
752, 756, 762, 770, 772, 774, 787, 791,
799, 800, 814, 821, 836, 828, 829, 830,
833, 862*872, 877*892, 900, 901, 906,
907, 912, 926, 934, 968, 970, 977, 980, 982,
993, 994, 1012, 1032, 1043, 1064, 1068,
1074, 1075, 1079, 1086, 1093, 1101, 1105,
1109, 1113, 1119, 1236, 1318, 1315, 1155,
1160, 1174, 1176, 1184, 1191, 1194, 1198,
1211, 1236, 1257, 1288, 1293, 1302, 1303,
1345, 1348, 1349, 1355, 1363, 1370, 1373.
Poiriot, Hercule: 1283.
\pool_file: 47, 50, 51*52, 53*.
\pool_name: 11*51*.
\pool_pointer: 38, 39, 45, 46, 59, 60, 69, 70,
264, 407, 464, 465, 470, 513, 519*602,
638, 929, 934, 1368, 1380*.
\pool_pbr: 38, 39, 41, 42, 43, 44, 47, 52, 58,
70, 198, 260, 464, 465, 470, 516*525, 617,
1309, 1310, 1332*1334, 1339*1368.
\pool_size: 11*38, 42, 52, 58, 198, 525, 1310,
1334, 1339*1368.
\pop: 584, 585, 586, 590, 601, 608, 642.
\pop_alignment: 772, 800.
\pop_input: 322, 324, 329.
\pop_stack: 910, 911.
\pop_nested: 217, 796, 799, 812, 816*1026, 1086,
1096, 1100, 1119, 1145, 1168, 1184, 1206.
\positive: 107.
\post: 583, 585, 586, 590, 591, 642.
\post_break: 145, 175, 195, 202, 206, 840,
858, 882, 884, 916, 1119.
\post_disc_break: 877*881, 884.
\righthyphenmin primitive: 238
\righthyphenmin_code: 236, 237, 238.
right_mode: 687, 690, 696, 698, 725, 728, 760, 761, 762, 1164, 1188, 1191.
right_ptr: 605, 606, 607, 615.
right_skip: 245, 287, 880, 881.
\rightskip primitive: 226
\rightskip_code: 245, 225, 226, 881, 886.
right1: 555, 556, 607, 610, 616.
right2: 555, 510.
right3: 555, 610.
right_t: 555, 610.
\rightlink: 124, 125, 126, 127, 129, 130, 131, 132, 145, 149, 164, 169, 772, 819, 821, 1311, 1312.
\romanprimitive: 468
\romannumeral_code: 468, 469, 471, 472.
rnd: 3, 114, 186, 625, 634, 809, 1125.
rnd_decimals: 103, 105, 452.
\rerver: 124, 125, 126, 127, 128, 129, 130, 131, 132, 164, 169, 1311, 1312.
\rhtd: 906, 907, 910, 911, 1033, 1035, 1040.
rule_up: 592, 622, 624, 636, 631, 633, 635.
rule_down: 592, 622, 624, 626, 631, 633, 634, 636.
\rule_node_size: 138, 139, 202, 206.
\rule_save: 800, 804.
rules_aligning_characters: 589.
Runaway...: 306.
s: 45, 46, 58, 59, 60, 62, 63, 93, 94, 95, 103.
108, 125, 130, 147, 177, 178, 204, 284.
389, 407, 473, 482, 529, 530, 560, 638.
645, 649, 668, 688, 699, 706, 720, 726.
738, 791, 800, 830, 862*877*901, 934.
966, 985, 1012, 1060, 1061, 1123, 1138, 1198, 1296, 1257, 1349, 1355.
save_command_ptr: 498, 500, 509.
save_context_ptr: 774, 777.
save_curr_val: 450, 455.
save_forafter: 250, 1271.
save_index: 268, 274, 276, 280, 282.
save_fink: 830, 857.
save_loc: 619, 629.
save_size: 11, 111, 271, 273, 1334.
save_sphiopskip: 1012, 1014.
save_style: 726, 794.
save_type: 268, 274, 276, 280, 282.
save: 610, 623, 628, 662, 632, 636, 637.
save_vbadness: 1012, 1017.
save_vfuzz: 1012, 1017.
save_warning_index: 389.
saved: 274, 645, 804, 1083, 1086, 1099, 1100, 1117, 1119, 1142, 1153, 1168, 1172, 1174, 1186, 1194.
scaled: 1258.
scaled_base: 247, 249, 251, 1224, 1237.
scan_box: 1073, 1084, 1241.
scan_charnum: 414, 434, 935, 1030, 1038, 1123, 1124, 1151, 1154, 1224, 1232.
scan_delimiter: 1160, 1163, 1182, 1183, 1191, 1192.
scan_dimen: 410, 440, 447, 448, 461, 462, 1061.
scan_rightbit_int: 415, 420, 427, 433, 505, 1079, 1082, 1099, 1110, 1224, 1226, 1227, 1237, 1241, 1247, 1296.
scan_fifteen_bit_int: 436, 1151, 1154, 1165, 1224.
scan_file_name: 265, 334, 526, 527, 537*.
s2: 1257, 1275, 1351.
scan_font_ident: 415, 426, 471, 517, 578, 1234, 1253.
scan_four_bit_int: 455, 501, 577, 1234, 1275, 1350.
scan_glue: 410, 461, 782, 1060, 1228, 1238.
scan_left_brace: 403, 473, 645, 785, 934, 960, 1025, 1099, 1117, 1119, 1153, 1172, 1174.
scan_math: 1150, 1151, 1158, 1163, 1165, 1176.
scan_optionals_eq: 405, 782, 1224, 1226, 1228, 1232, 1234, 1236, 1241, 1243, 1244, 1245, 1246, 1247, 1248, 1253, 1257, 1275, 1351.
scan_rule_spec: 463, 1056, 1084.
scan_something_internal: 409, 410, 413, 432, 440, 449, 451, 455, 461, 465.
scan_spec: 645, 768, 774, 1071, 1083, 1167.
scan_twenty_seven_bit_int: 437, 1151, 1154, 1160.
scanned_result: 413, 414, 415, 418, 422, 425, 426, 428.
scanned_result_end: 413.
\scriptfont primitive: 1230.
\scriptlist: 689, 695, 698, 731, 1174.
\scriptfont primitive: 1230.
\scriptlist: 689, 695, 698, 731, 1174.
\scriptsize: 699, 756, 1195, 1230.
\scriptstyle: 688, 694, 731, 1169.
\scriptscriptstyle primitive: 1169.
\scriptsize: 699, 756, 1195, 1230.
\scriptstyle: 688, 694, 702, 703, 731, 756, 762, 766, 1169.
\scriptstyle: 688, 694, 702, 703, 731, 756, 762, 766, 1169.

scroll_mode: 71, 73, 84, 86, 93, 530, 1262, 1263, 1281.
\scrollmode primitive: 1262.
search_mem: 165, 172, 255, 1339*. 
second_indent: 847, 848, 849, 889.
second_width: 847, 848, 849, 850, 889.
Sedgwick, Robert: 2*
see the transcript file...: 1335.
sel: 1380*.
semi_simple_group: 209, 1063, 1065, 1068, 1069.
serial: 821, 845, 846, 856.
set(aux): 209, 413, 416, 417, 418, 1210, 1242.
set(box): 209, 265, 266, 1210, 1241.
\setbox primitive: 265.
set_box: 26, 77, 1241, 1270.
set_box_dimen: 209, 413, 416, 417, 1210, 1242.
set_break_width_to_background: 837.
set_char: 585, 586, 620.
set_conversion: 458.
set_conversion_end: 458.
set_curlang: 234, 960, 1091, 1200.
set_curlang: 908, 910, 911.
set_font: 209, 413, 553, 577, 1210, 1217, 1257, 1261.
set glue ratio one: 100, 664, 676, 810, 811.
set glue ratio zero: 100, 136, 657, 658, 664, 672, 673, 676, 810, 811.
set_height_zero: 970.
set_inter: 209, 1210, 1262, 1263, 1264.
\setlanguage primitive: 1344.
\setlanguage_code: 1344, 1346, 1348.
set_math: 1154, 1155.
set_page: 209, 413, 982, 983, 984, 1210, 1242.
set_page_int: 209, 413, 416, 417, 1210, 1242.
set_page: 987.
set_prevgraf: 209, 265, 266, 413, 1210, 1242.
set: 585, 586, 620.
set_shape: 209, 265, 266, 413, 1210, 1248.
sf: 585.
sf: 585.
sf: 585.
sf: 230, 232, 1034.
\texttt{sfcode} primitive: 1230
\texttt{sfcode base}: 220, 235, 1230, 1231, 1233.
\texttt{shape ref}: 210, 222, 275, 1070, 1248.
\texttt{shift case}: 1285, 1288
\texttt{shift down}: 743, 744, 745, 746, 747, 749, 751, 756, 757, 759.
\texttt{shift up}: 743, 744, 745, 746, 747, 749, 751, 756, 758, 759.
\texttt{ship out}: 392, 638, 644, 1023, 1075.
\texttt{ship out flag}: 1071, 1075.
\texttt{short display}: 173, 174, 175, 193, 663, 857, 1339*
\texttt{short read}: 109*110.
\texttt{short cut}: 447, 448
\texttt{short fall}: 830, 851, 852, 853.
\texttt{shorthand def}: 200, 1210, 1222, 1223, 1224.
\texttt{show primitive}: 1291.
\texttt{show activities}: 218, 1293.
\texttt{show box}: 180, 182, 198, 218, 219, 236, 638, 641, 663, 675, 986, 992, 1121, 1296, 1339*
\texttt{show box primitive}: 1291.
\texttt{show box breadth}: 236, 1339*
\texttt{showbox breadth primitive}: 238.
\texttt{show box breadth code}: 236, 237, 238.
\texttt{show box code}: 1291, 1292, 1293.
\texttt{show box depth}: 236, 1339*
\texttt{show box depth primitive}: 238.
\texttt{show box depth code}: 236, 237, 238.
\texttt{show code}: 1291, 1293.
\texttt{show context}: 54, 78, 82, 88, 310, 311, 318, 530, 535, 537*.
\texttt{show cur end char}: 290, 367, 1031.
\texttt{show eq}: 252, 284.
\texttt{show info}: 692, 693.
\texttt{show lists code}: 1291, 1292, 1293.
\texttt{show lists primitive}: 1291.
\texttt{show node list}: 173, 176, 180, 181, 182, 195, 198, 233, 690, 692, 693, 695, 1339*.
\texttt{showthe primitive}: 1291.
\texttt{show the code}: 1291, 1292.
\texttt{show whatever}: 1290, 1293.
\texttt{shownode}: 213, 215, 299.
\texttt{shrink}: 150, 151, 164, 178, 431, 462, 625, 634, 636, 671, 716, 809, 825, 827, 838, 868, 976, 1004, 1009, 1042, 1044, 1148, 1229, 1239, 1240.
\texttt{shrink order}: 150, 164, 178, 462, 625, 634, 656, 671, 716, 809, 825, 826, 976, 1004, 1009, 1148, 1239.
\texttt{shrinking}: 135, 186, 619, 629, 664, 676, 809, 810, 811, 1148.
\texttt{si}: 38, 42, 69, 951, 964, 1310.
\texttt{SIGINT}: 1381*
\texttt{signal}: 1381*
\texttt{simple group}: 269, 1063, 1068.
Single-character primitives: 267
\texttt{\textbackslash -}: 1114
\texttt{\textbackslash /}: 265
\texttt{\textbackslash :}: 265.
\texttt{single base}: 222, 262, 263, 264, 354, 374, 442, 1257, 1289.
\texttt{skew char}: 426, 549, 552, 576, 741, 1253, 1322, 1323.
\texttt{\textbackslash skewchar} primitive: 1254.
\texttt{skip}: 224, 427, 1009.
\texttt{\textbackslash skip} primitive: 411.
\texttt{skip base}: 224, 227, 229, 1224, 1237.
\texttt{skip blanks}: 303, 344, 345, 347, 349, 354.
\texttt{skip byte}: 545, 557, 741, 752, 753, 909, 1039.
\texttt{skew code}: 1038, 1059, 1060.
\texttt{\textbackslash skipdef} primitive: 1222.
\texttt{skip def code}: 1222, 1223, 1224.
\texttt{skip line}: 336, 493, 494.
\texttt{skipping}: 305, 306, 336, 494.
\texttt{slant}: 547, 558, 575*, 1123, 1125.
\texttt{slant code}: 547, 558.
\texttt{slow print}: 60, 61, 63, 84*, 518, 536, 537*581, 642, 1261, 1280, 1283, 1328, 1333*1339*.
\texttt{small char}: 683, 691, 697, 706, 1160.
\texttt{small fam}: 683, 691, 697, 706, 1160.
\texttt{small node size}: 141, 144, 145, 147, 152, 153, 156, 158, 202, 206, 655, 721, 903, 910, 914, 1037, 1100, 1101, 1357, 1358, 1376, 1377.
\texttt{so}: 38, 45, 59, 60, 69, 70, 264, 407, 464, 519*603, 617, 766, 931, 953, 955, 956, 959, 963, 1309, 1368.
\texttt{Sorry, I can't find...}: 524*
\texttt{sort avail}: 131, 1311.
59, 60, 69, 70, 256, 260, 264, 407, 517, 519*603, 617, 765, 929, 931, 934, 941, 1309, 1310, 1368, 1380*

\strtoks: 464, 465, 470.
\stretch: 150, 151, 164, 178, 431, 462, 625, 634, 636, 671, 716, 809, 827, 838, 868, 976, 1004, 1009, 1042, 1044, 1148, 1229, 1239, 1240.
\stretch_order: 130, 164, 178, 462, 625, 634, 656, 671, 716, 809, 827, 838, 868, 976, 1004, 1009, 1148, 1239.

\stretching: 135, 625, 634, 658, 673, 809, 810, 811, 1148.

\string pool: 47, 1308.
\string primitive: 468.
\string code: 468, 469, 471, 472.
\string vacancies: 11*52.
\style: 726, 760, 761, 762.
\style node: 160, 588, 690, 698, 730, 731, 761, 1169.
\style node size: 688, 689, 698, 763.
\sub box: 651, 687, 692, 698, 720, 734, 735, 737, 758, 749, 754, 1076, 1093, 1168.
\sub drop: 700, 756.
\sub mark: 207, 294, 298, 347, 1046, 1175.
\sub list: 651, 683, 692, 720, 742, 754, 1181, 1185, 1186, 1187.
\sub style: 702, 750, 757, 759.
\sub sup: 1175, 1176.
\subscr: 651, 683, 686, 687, 690, 696, 698, 738, 742, 749, 750, 751, 752, 753, 754, 755, 756, 757, 759, 1151, 1163, 1165, 1175, 1176, 1177, 1186.
\subscripts: 754, 1175.
\subscript: 700, 757.
\subsuccess: 93, 94, 95, 1304.
\sup drop: 700, 756.
\sup mark: 207, 294, 298, 344, 355, 1046, 1175, 1176, 1177.
\sup style: 702, 750, 758.
superscripts: 754, 1175.
super: 651, 683, 686, 687, 690, 696, 698, 738, 742, 750, 751, 752, 753, 754, 756, 758, 1151, 1163, 1165, 1175, 1176, 1177, 1186.
sup1: 700, 758.
sup2: 700, 758.
sup3: 700, 758.
sw: 560, 571, 575*.
synch_h: 616, 620, 624, 628, 633, 637, 1368.
synch_v: 616, 620, 624, 628, 632, 633, 637, 1368.
sys_day: 241*246, 536.
sys_month: 241*246, 536.
sys_time: 241*246, 536.
sys_year: 241*246, 536.
system dependencies: 2*3, 4*9*10*11*12, 19, 21, 23*26, 27*28, 33*34*35, 36*38, 49, 56, 59, 61, 72, 81, 84*96, 109*110, 112*113, 161, 186, 241*304, 313, 328, 485, 511, 512, 513, 514*515, 516*517, 518, 519*520, 521*523*525, 538, 557, 564, 591, 595, 597*798, 1306, 1331, 1332*, 1333*1338*1340, 1380*1381*1382*.
sysutils: 241*.
sysutils: 241*.
s1: 82, 88.
s2: 82, 88.
s3: 82, 88.
s4: 82, 88.

\tab_open: 38*37*.
\tab_open_out: 33*1332*.
\tab_mark: 207, 294, 298, 342, 347, 780, 1126.
\tab_skip: 224.
\tabskip primitive: 226.
\tabskip code: 224, 225, 226, 778, 782, 786, 793, 795, 809.
\tab_token: 289, 1128.
tag: 543, 544, 554.
TEX area: 514*537*
TEX fonts area: 514*563.
TEX format default: 520, 521*523*
TEX fonts: 514*
TEX formats: 11*521*
TEX inputs: 514*
texput: 35, 334, 1257.

Text line contains:... 346.
text_char: 19, 20, 47.
\textfont primitive: 1230
\text_math_font primitive: 1230
\text_dist primitive: 689, 695, 698, 731, 1174.
text_size: 699, 703, 732, 762, 1195, 1199.
text_style: 688, 694, 703, 731, 737, 744, 745, 746, 748, 749, 758, 762, 1169, 1194, 1196.
\text_style primitive: 1169
TEX82: L, 99.
TFM files: 539.
tfm_file: 339, 560, 563, 564.
\TFToPL: 561.
That makes 100 errors:... 82.
the: 210, 265, 266, 366, 367, 478.
The following... deleted: 641, 992, 1121.
\the primitive: 265.
the_toks: 465, 466, 467, 478, 1297.
\thickmu_skip primitive: 224
\thickmu_skip primitive: 226
\thickmu_skip_code: 224, 225, 226, 766.
thickness: 683, 697, 725, 743, 744, 746, 747, 1182
\thinmu_skip: 224
\thinmu_skip primitive: 226
\thinmu_skip_code: 224, 225, 226, 229, 766.
This can't happen: 95.
align: 800.
\copying primitive: 206.
\curlevel primitive: 281.
disc1: 841.
disc2: 842.
disc3: 870.
disc4: 871.
display: 1200.
endv: 791.
\ext1: 1348.
\ext2: 1357.
\ext3: 1358.
\ext4: 1373.
if: 497.
line breaking: 877*
mist1: 728.
mist2: 754.
mist3: 761.
mist4: 766.
page: 1000.
paragraph: 866.
prefix: 1211.
pruning: 968.
right: 1185.
right_brace: 1068.
vcnter: 736.
vertbreak: 973.
vlistout: 630.
vpack: 669.
256 spans: 798.
this_box: 619, 624, 625, 629, 633, 634.
this_if: 498, 501, 503, 505, 506.
three_codes: 645.
threshold: 823, 851, 854, 863.
Tight \hbox...: 667.
Tight \vbox...: 678.
tight_fit: 817, 819, 830, 833, 834, 836, 853.
time: 236, 241*617.
\time primitive: 238.
\time_code: 236, 237, 238.
tini: 8.
to: 645, 1082, 1225.
token: 289.
token_ref_count: 200, 303, 291, 473, 482, 979.
token_show: 295, 296, 323, 401, 1279, 1284, 1297, 1370.
toks: 230.
\toks primitive: 265.
\toksdef primitive: 1222.
\toks_def_code: 1222, 1224.
toks_register: 200, 265, 266, 413, 415, 1210, 1226, 1227.
tolerance: 236, 240, 828, 863.
\tolerance primitive: 238.
tolerance_code: 236, 237, 238.
Too many \{s: 1068.
too_small: 1303, 1306.
top: 546.
top_bot_mark: 210, 296, 366, 367, 384, 385, 386.
top_edge: 629, 636.
top_mark: 382, 383, 1012.
\topmark primitive: 384.
top_mark_code: 382, 384, 386, 1335.
top_skip: 224.
\topskip primitive: 225.
top_skip_code: 224, 225, 226, 1001.
total_errors: 819, 845, 846, 855, 864, 874, 875.
total_height: 986.
total_math_paras: 701, 1195.
total_mathy_paras: 700, 1195.
total_pages: 392, 593, 617, 640, 642.
total_shrink: 646, 650, 656, 664, 665, 666, 667, 671, 676, 677, 786, 1201.
total_stretch: 646, 650, 656, 658, 659, 660, 671, 673, 674, 796.
Tribb Pardo, Luis Isidoro: 2*
tracing_commands: 236, 367, 498, 509, 1031.
\tracingcommands primitive: 245.
tracing_commands_code: 236, 237, 238.
tracing_lost_chars: 236, 581.
\tracinglostchars primitive: 238.
tracing_lost_chars_code: 236, 237, 238.
tracing_macros: 236, 323, 389, 400.
\tracingmacros primitive: 238.
tracing_macros_code: 236, 237, 238.
tracing_online: 236, 245, 1293, 1298.
\tracingonline primitive: 245.
tracing_online_code: 236, 237, 238.
tracing_output: 236, 638, 641.
\tracingoutput primitive: 238.
tracing_output_code: 236, 237, 238.
tracing_pages: 236, 987, 1005, 1010.
\tracingpages primitive: 238.
tracing_pages_code: 236, 237, 238.
tracing_paragraphs: 236, 826, 845, 855, 863.
\tracingparagraphs primitive: 238.
tracing_paragraphs_code: 236, 237, 238.
tracing_restores: 236, 238.
\tracingrestores primitive: 238.
tracing_restores_code: 236, 237, 238.
tracing_stats: 117, 236, 639, 1326, 1333*.
\tracingstats primitive: 238.
tracing_stats_code: 236, 237, 238.
Transcript written...: 1333*.
trap_zero glue: 1228, 1229, 1236.
trick_del: 54, 58, 315, 317.
trick_count: 54, 58, 315, 316, 317.
Trickev, Howard Wellington: 2*
trie_back: 950, 954, 956.
trie_c: 947, 948, 951, 953, 955, 956, 959, 963, 964.
trie_char: 920, 921, 923, 958, 959.
trie_file: 958, 959.
trie_hash: 947, 948, 949, 950, 952.
trie_j: 947, 948, 949, 957, 959, 960, 963, 964.
trie_link: 920, 921, 923, 950, 952, 953, 954, 955, 956, 958, 959.
trie_max: 950, 952, 954, 958, 1324, 1325.
trie_min: 950, 952, 953, 956.
trie_node: 948, 949.
trie_not_ready: 891, 950, 951, 960, 966, 1324, 1325.
trie_op: 947, 948, 959, 963, 964.
trie_op_file: 920, 921, 923, 924, 943, 958, 959.
trie_op_hash: 943, 944, 945, 946, 948, 952.
trie_op_lang: 943, 944, 945, 952.
trie_op_ptr: 943, 944, 945, 946, 1324, 1325.
trie_op_size: 11*920, 943, 944, 946, 1324, 1325.
trie_op_val: 943, 944, 945, 952.
trie_pack: 957, 966.
trie_ptr: 947, 951, 952, 964.
trie_ref: 950, 952, 953, 956, 957, 959, 963, 964.
trie_size: 11*920, 948, 950, 952, 954, 964, 1325.
trie_taken: 920, 952, 953, 954, 956.
trie_used: 943, 944, 945, 946, 1324, 1325.
true: 453.
\texttt{w}: 585
\texttt{x}: 100, 105, 106, 107, 387, 600, 649, 668, 706, 720, 726, 735, 737, 738, 743, 749, 756, 1123, 1302, 1303
\texttt{x_height}: 547, 558, 559, 738, 1123.
\texttt{x_height_code}: 547, 558.
\texttt{x_leaders}: 149, 190, 627, 1071, 1072.
\texttt{\textbackslashleaders primitive}: 1071.
\texttt{x_over_n}: 106, 703, 716, 717, 986, 1008, 1009, 1010, 1240.
\texttt{x_token}: 364, 381, 478, 1038, 1152.
\texttt{xch}: 20, 21, 23, 24, 38, 49, 58, 519, 1380.
\texttt{xclause}: 16.
\texttt{\textbackslashdef primitive}: 1308.
\texttt{xreqlevel}: 233, 254, 268, 278, 279, 283, 1304.
\texttt{xm_over_j}: 107, 455, 457, 458, 368, 716, 1044, 1260.
\texttt{xord}: 20, 24, 31, 36, 52, 53, 523, 525.
\texttt{xrand}: 473, 477, 479.
\texttt{xmay}: 208, 1290, 1291, 1292.
\texttt{xspace\_skip}: 224, 1043.
\texttt{xspaceskip primitive}: 226.
\texttt{xspace\_skip\_code}: 224, 225, 226, 1043.
\texttt{xxx1}: 585, 586, 1368.
\texttt{xxx2}: 585.
\texttt{xxx3}: 585.
\texttt{xxx4}: 585, 586, 1368.
\texttt{xd0}: 585, 586, 604, 609.
\texttt{xd1}: 585, 586, 607.
\texttt{xd2}: 585.
\texttt{xd3}: 585.
\texttt{xd4}: 585.
\texttt{y}: 105, 706, 726, 735, 737, 738, 743, 749, 756
\texttt{y\_here}: 608, 609, 611, 612, 613.
\texttt{y\_OK}: 608, 609, 612.
\texttt{y\_seen}: 611, 612.
\texttt{year}: 236, 241, 617, 1328.
\texttt{\textbackslashyear primitive}: 238.
\texttt{\textbackslashyear\_code}: 236, 237, 238.
You already have nine...: 476.
You can't \textbackslashinsert255: 1099.
You can't \textbackslashdump: 1304.
You can't use \textbackslashhrule: 1095.
You can't use \textbackslashlong: 1213.
You can't use a prefix with x: 1212.
You can't use x after...: 428, 1237.
You can't use x in y mode: 1049.
You have to increase POOLSIZE: 52.
You want to edit file x: 84
\texttt{you\_can't}: 1049, 1050, 1080, 1106.
\texttt{yy}: 241.*
\texttt{yy\_OK}: 608, 609, 610, 612.
Accumulate the constant until \( \text{cur}_j \) is not a suitable digit 445 Used in section 444.
Add the width of node \( s \) to \( \text{act}_w \) 871 Used in section 869.
Add the width of node \( s \) to \( \text{break}_w \) 842 Used in section 840.
Add the width of node \( s \) to \( \text{disc}_w \) 870 Used in section 869.
Adjust for the magnification ratio 457 Used in section 453.
Adjust for the setting of \( \text{globaldefs} \) 1214 Used in section 1211.
Adjust \( \text{shift}_{up} \) and \( \text{shift}_{down} \) for the case of a fraction line 746 Used in section 743.
Adjust \( \text{shift}_{up} \) and \( \text{shift}_{down} \) for the case of no fraction line 745 Used in section 743.
Advance \( \text{cur}_p \) to the node following the present string of characters 867 Used in section 866.
Advance past a whatstaf node in the line break loop 1362 Used in section 866.
Advance past a whatstaf node in the pre-hyphenation loop 1363 Used in section 896.
Advance \( r; \) \text{goto found} if the parameter delimiter has been fully matched, otherwise \text{goto continue} 304 Used in section 392.
Allocate entire node \( p \) and \text{goto found} 129 Used in section 127.
Allocate from the top of node \( p \) and \text{goto found} 128 Used in section 127.
Apologize for inability to do the operation now, unless \text{unskip} follows non-glue 1106 Used in section 1105.
Apologize for not loading the font, \text{goto done} 567 Used in section 566.
Append a ligature and/or kern to the translation; \text{goto continue} if the stack of inserted ligatures is nonempty 910 Used in section 906.
Append a new leader node that uses \text{cur}_b \) 1078 Used in section 1075.
Append a new letter or a hyphen level 962 Used in section 961.
Append a new letter or hyphen 937 Used in section 935.
Append a normal inter-word space to the current list, then \text{goto big Switch} 1011 Used in section 1030.
Append a penalty node, if a nonzero penalty is appropriate 890 Used in section 880.
Append an insertion to the current \text{page} and \text{goto contribute} 1008 Used in section 1000.
Append any \text{new hlist} entries for \( q \), and any appropriate penalties 767 Used in section 760.
Append box \text{cur}_b \) to the current list, shifted by \text{box context} 1076 Used in section 1075.
Append character \text{cur}_c \) and the following characters (if any) to the current hlist in the current font; \text{goto reswitch} when a non-character has been fetched 1034 Used in section 1030.
Append characters of hu[j...k] to \text{major_tail}, advancing \( j \) 917 Used in section 916.
Append inter-element spacing based on \text{r_type} and \( t \) 766 Used in section 760.
Append tabskip glue and an empty box to list \( u \), and update \( s \) and \( t \) as the prototype nodes are passed 809 Used in section 808.
Append the accent with appropriate kerns, then set \( p \rightarrow q \) 1125 Used in section 1123.
Append the current tabskip glue to the preamble list 778 Used in section 777.
Append the display and perhaps also the equation number 1204 Used in section 1199.
Append the glue or equation number following the display 1205 Used in section 1199.
Append the glue or equation number preceding the display 1203 Used in section 1199.
Append the new box to the current vertical list, followed by the list of special nodes taken out of the box by the packager 888 Used in section 880.
Append the value \( n \) to list \( p \) 938 Used in section 937.
Assign the values depth_threshold \( \leq \) show_box_depth and breadth_max \( \leq \) show_box_breadth 236 Used in section 198.
Assignments 1217, 1218, 1221, 1224, 1225, 1226, 1228, 1232, 1234, 1235, 1241, 1242, 1248, 1252, 1253, 1256, 1264 Used in section 1211.
Attach list \( p \) to the current list, and record its length; then finish up and return 1120 Used in section 1119.
Attach the limits to \( y \) and adjust \( \text{height}(v), \text{depth}(v) \) to account for their presence 751 Used in section 750.
(Back up an outer control sequence so that it can be reread 337) Used in section 336.
(Basic printing procedures 57, 58, 59, 60, 62, 63, 64, 65, 262, 263, 518, 609, 1355) Used in section 4.8.
(Break the current page at node $p$, put it in box 255, and put the remaining nodes on the contribution list 1017) Used in section 1014.
(Break the paragraph at the chosen breakpoints, justify the resulting lines to the correct widths, and append them to the current vertical list 876*) Used in section 815.
(Calculate the length, $l$, and the shift amount, $s$, of the display lines 1149) Used in section 1145.
(Calculate the natural width, $w$, by which the characters of the final line extend to the right of the reference point, plus two ems; or set $w \leftarrow \text{max_dimen}$ if the non-blank information on that line is affected by stretching or shrinking 1146) Used in section 1145.
(Call the packaging subroutine, setting $just_box$ to the justified box 889) Used in section 880.
(Call $try\_break$ if $cur\_p$ is a glue node; then advance $cur\_p$ to the next node of the paragraph that could possibly be a legal breakpoint 866) Used in section 863.
(Carry out a ligature replacement, updating the cursor structure and possibly advancing $j$; $goto$ $continue$ if the cursor doesn’t advance, otherwise $goto$ $done$ 911) Used in section 909.
(Case statement to copy different types and set $words$ to the number of initial words not yet copied 206) Used in section 205.
(Cases for nodes that can follow a $bin\_node$ 733) Used in section 728.
(Cases for nodes that can appear in an mlist, after which we $goto$ $done\_with\_node$ 730) Used in section 728.
(Cases of $flush\_node\_list$ that arise in mlists only 668) Used in section 202.
(Cases of $handle\_right\_brace$ where a $right\_brace$ triggers a delayed action 1085, 1100, 1118, 1132, 1133, 1168, 1173, 1186) Used in section 1068.
(Cases of $main\_control$ that are for extensions to $\TeX$ 1347) Used in section 1045.
(Cases of $main\_control$ that are not part of the inner loop 1045) Used in section 1030.
(Cases of $main\_control$ that build boxes and lists 1056, 1057, 1063, 1067, 1073, 1090, 1092, 1094, 1097, 1102, 1104, 1109, 1112, 1116, 1122, 1126, 1130, 1134, 1137, 1140, 1150, 1154, 1158, 1162, 1164, 1167, 1171, 1175, 1180, 1190, 1193) Used in section 1045.
(Cases of $main\_control$ that don’t depend on $mode$ 1210, 1268, 1271, 1274, 1276, 1285, 1290) Used in section 1045.
(Cases of $print\_cmd\_chr$ for symbolic printing of primitives 227, 231, 239, 249, 266, 335, 377, 385, 412, 417, 469, 488, 492, 781, 984, 1033, 1059, 1072, 1089, 1108, 1115, 1143, 1157, 1170, 1179, 1189, 1209, 1220, 1223, 1231, 1251, 1255, 1261, 1263, 1273, 1278, 1287, 1292, 1295, 1346) Used in section 298.
(Cases of $show\_node\_list$ that arise in mlists only 600) Used in section 183.
(Cases where character 3 is ignored 345) Used in section 344.
(Change buffered instruction to $y$ or $w$ and $goto$ $found$ 613) Used in section 612.
(Change buffered instruction to $z$ or $x$ and $goto$ $found$ 614) Used in section 612.
(Change current mode to $-vmode$ for $\text{\texttt{align}}$, $-hmode$ for $\text{\texttt{align}}$ 775) Used in section 774.
(Change discretionary to compulsory and set $disc\_break\leftarrow true$ 882) Used in section 881.
(Change font $dvi\_f$ to $f$ 621) Used in section 620.
(Change state if necessary, and $goto$ $switch$ if the current character should be ignored, or $goto$ $reswitch$ if the current character changes to another 344) Used in section 343.
(Change the case of the token in $p$, if a change is appropriate 1289) Used in section 1288.
(Change the current style and $goto$ $delete\_q$ 763) Used in section 761.
(Change the interaction level and $return$ 864) Used in section 844.
(Change this node to a style node followed by the correct choice, then $goto$ $done\_with\_node$ 731) Used in section 730.
(Character $k$ cannot be printed 49) Used in section 48.
(Character $s$ is the current new-line character 244) Used in sections 58 and 59.
(Check flags of unavailable nodes 170) Used in section 167.
(Check for charlist cycle 570) Used in section 509.
Check for improper alignment in displayed math 776 
Check if node $p$ is a new champion breakpoint; then $\text{goto done}$ if $p$ is a forced break or if the page-so-far is already too full 974 
Check if node $p$ is a new champion breakpoint; then $\text{goto done}$ if it is time for a page break, prepare for output, and either fire up the user's output routine and $\text{return}$ or ship out the page and $\text{goto done}$ 1005 
Check single-word avail list 168 
Check that another $\$ follows 1197 
Check that the necessary fonts for math symbols are present; if not, flush the current math lists and set danger $\leftarrow$ true 1195 
Check that the nodes following $\text{hb}$ permit hyphenation and that at least $L_{\text{hyf}} + r_{\text{hyf}}$ letters have been found, otherwise $\text{goto done1}$ 899 
Check the "constant" values for consistency 14, 111, 290, 522, 1249 
Check the pool check sum 53* 
Check variable-size avail list 109 
Clean up the memory by removing the break nodes 805 
Clear dimensions to zero 
Clear off top level from $\text{save_stack}$ 282 
Close the format file 1329 
Coerce glue to a dimension 451 
Compiler directives 9* 
Complain about an undefined family and set $\text{cur_f} \leftarrow$ null 723 
Complain about an undefined macro 370 
Complain about missing $\text{\endname}$ 373 
Complain about unknown unit and $\text{goto done2}$ 459 
Complain that $\text{\the can't do this};$ give zero result 428 
Complain that the user should have said $\text{\mathaccent}$ 1166 
Compleat the incompleat node 1185 
Complete a potentially long $\text{\show}$ command 1288 
Compute result of $\text{multiply or divide}$, put it in $\text{cur_val}$ 1240 
Compute result of $\text{register or advance}$, put it in $\text{cur_val}$ 1238 
Compute the amount of skew 741 
Compute the badness, $b$, of the current page, using $\text{awfulbad}$ if the box is too full 1007 
Compute the badness, $b$, using $\text{awfulbad}$ if the box is too full 975 
Compute the demerits, $d$, from $r$ to $\text{cur_p}$ 859 
Compute the discretionary break_width values 840 
Compute the hash code $h$ 261 
Compute the magic offset 765 
Compute the minimum suitable height, $w$, and the corresponding number of extension steps, $n$; also set width($b$) 714 
Compute the new line width 850 
Compute the register location $l$ and its type $p_l$ but return if invalid 1237 
Compute the sum of two glue specs 1239 
Compute the trie op code, $v$, and set $l \leftarrow 0$ 965 
Compute the values of break_width 837 
Consider a node with matching width; $\text{goto found}$ if it's a hit 612 
Consider the demerits for a line from $r$ to $\text{cur_p}$; deactivate node $r$ if it should no longer be active; then $\text{goto continue}$ if a line from $r$ to $\text{cur_p}$ is infeasible, otherwise record a new feasible break 851 
Constants in the outer block 11* 
Construct a box with limits above and below it, skewed by $\Delta$ 750 

Declare procedures needed in

Construct a sub/superscript combination box $x$, with the superscript offset by $delta$ 759

Used in section 756.

Construct a subscript box $x$ when there is no superscript 757

Used in section 756.

Construct a superscript box $x$ 758

Used in section 756.

Construct a vlist box for the fraction, according to $shift_{up}$ and $shift_{down}$ 747

Used in section 743.

Construct an extensible character in a new box $b$, using recipe $rem_{byte}(q)$ and font $f$ 713

Used in section 710.

Contribute an entire group to the current parameter 399

Used in section 392.

Contribute the recently matched tokens to the current parameter, and $goto$ continue if a partial match is still in effect; but abort if $s = null$ 397

Used in section 392.

Convert a final $bin_noad$ to an $ord_noad$ 729

Used in sections 726 and 728.

Convert $cur_{val}$ to a lower level 429

Used in section 413.

Convert math glue to ordinary glue 732

Used in section 730.

Convert $nucleus(q)$ to an hlist and attach the sub/superscripts 754

Used in section 728.

Copy the tabskip glue between columns 795

Used in section 791.

Copy the templates from node $cur_{loop}$ into node $p$ 794

Used in section 793.

Copy the token list 466

Used in section 465.

Create a character node $p$ for $nucleus(q)$, possibly followed by a kern node for the italic correction, and set $delta$ to the italic correction if a subscript is present 755

Used in section 754.

Create a character node $q$ for the next character, but set $q ← null$ if problems arise 1124

Used in section 1123.

Create a new glue specification whose width is $cur_{val}$; scan for its stretch and shrink components 462

Used in section 461.

Create a page insertion node with $subtype(r) = q_i(n)$, and include the glue correction for box $n$ in the current page state 1009

Used in section 1008.

Create an active breakpoint representing the beginning of the paragraph 864

Used in section 863.

Create and append a discretionary node as an alternative to the unhyphenated word, and continue to develop both branches until they become equivalent 914

Used in section 913.

Create equal-width boxes $x$ and $z$ for the numerator and denominator, and compute the default amounts $shift_{up}$ and $shift_{down}$ by which they are displaced from the baseline 744

Used in section 743.

Create new active nodes for the best feasible breaks just found 836

Used in section 835.

Create the $format_{ident}$, open the format file, and inform the user that dumping has begun 1328

Used in section 1302.

Current $mem$ equivalent of glue parameter number $n$ 224

Used in sections 152 and 154.

Deactivate node $r$ 860

Used in section 851.

Declare action procedures for use by $main_{control}$ 1043, 1047, 1049, 1050, 1051, 1054, 1060, 1061, 1064, 1069, 1070, 1075, 1079, 1084, 1086, 1091, 1093, 1095, 1096, 1099, 1101, 1103, 1105, 1110, 1113, 1117, 1119, 1123, 1127, 1129, 1131, 1135, 1136, 1138, 1142, 1151, 1155, 1159, 1160, 1163, 1165, 1172, 1174, 1176, 1181, 1191, 1194, 1300, 1211, 1270, 1275, 1279, 1288, 1293, 1302, 1348, 1376

Used in section 1030.

Declare math construction procedures 734, 735, 736, 737, 738, 743, 749, 752, 756, 762

Used in section 726.

Declare procedures for preprocessing hyphenation patterns 944, 948, 949, 953, 957, 993, 999, 960, 966

Used in section 942.

Declare procedures needed for displaying the elements of mlists 601, 692, 694

Used in section 179.

Declare procedures needed in $do_{extension}$ 1349, 1350

Used in section 1348.

Declare procedures needed in $hlist_{out}$, $vlist_{out}$ 1368, 1370, 1373

Used in section 619.

Declare procedures that scan $font_{out}$, $font_{in}$ 577, 578

Used in section 409.
\begin{itemize}
  \item \textbf{Declare procedures that scan restricted classes of integers 433, 434, 435, 436, 437} \\
    Used in section 409.
  \item \textbf{Declare subprocedures for \texttt{line\_break} 826, 829, 877*, 895, 942} \\
    Used in section 815.
  \item \textbf{Declare subprocedures for \texttt{prefix\_command} 1215, 1229, 1236, 1243, 1244, 1245, 1246, 1247, 1257, \\
    1265*} \\
    Used in section 1211.
  \item \textbf{Declare subprocedures for \texttt{var\_delimiter} 709, 711, 712} \\
    Used in section 706.
  \item \textbf{Declare the function called \texttt{find\_list} 1184} \\
    Used in section 1174.
  \item \textbf{Declare the function called \texttt{open\_hint\_file} 524*} \\
    Used in section 1303.
  \item \textbf{Declare the function called \texttt{reconstitute} 906} \\
    Used in section 805.
  \item \textbf{Declare the procedure called \texttt{align\_peek} 785} \\
    Used in section 800.
  \item \textbf{Declare the procedure called \texttt{fire\_up} 1012} \\
    Used in section 994.
  \item \textbf{Declare the procedure called \texttt{get\_preamble\_token} 782} \\
    Used in section 774.
  \item \textbf{Declare the procedure called \texttt{handle\_right\_brace} 1068} \\
    Used in section 1000.
  \item \textbf{Declare the procedure called \texttt{init\_span} 787} \\
    Used in section 786.
  \item \textbf{Declare the procedure called \texttt{insert\_relax} 379} \\
    Used in section 366.
  \item \textbf{Declare the procedure called \texttt{macro\_call} 389} \\
    Used in section 366.
  \item \textbf{Declare the procedure called \texttt{print\_cmd\_chr} 298} \\
    Used in section 252.
  \item \textbf{Declare the procedure called \texttt{print\_skip\_param} 225} \\
    Used in section 179.
  \item \textbf{Declare the procedure called \texttt{restore\_brace} 284} \\
    Used in section 281.
  \item \textbf{Declare the procedure called \texttt{runaway} 306} \\
    Used in section 119.
  \item \textbf{Declare the procedure called \texttt{show\_token\_list} 292} \\
    Used in section 119.
  \item \textbf{Decry the invalid character and \texttt{goto restart} 346} \\
    Used in section 344.
  \item \textbf{Delete e = "0" tokens and \texttt{goto continue} 88} \\
    Used in section 84*.
  \item \textbf{Delete the page-insertion nodes 1019} \\
    Used in section 1014.
  \item \textbf{Destroy the t nodes following q, and make r point to the following node 883} \\
    Used in section 882.
  \item \textbf{Determine horizontal glue shrink setting, then \texttt{return or goto common\_ending 664} } \\
    Used in section 657.
  \item \textbf{Determine horizontal glue stretch setting, then \texttt{return or goto common\_ending 658} } \\
    Used in section 657.
  \item \textbf{Determine the displacement, d, of the left edge of the equation, with respect to the line size z,} \\
    assuming that l = false 1202 \\
    Used in section 1199.
  \item \textbf{Determine the shrink order 665} \\
    Used in sections 664, 676, and 796.
  \item \textbf{Determine the stretch order 659} \\
    Used in sections 658, 673, and 796.
  \item \textbf{Determine the value of height(r) and the appropriate glue setting; then \texttt{return or goto common\_ending 672} } \\
    Used in section 668.
  \item \textbf{Determine the value of width(r) and the appropriate glue setting; then \texttt{return or goto common\_ending 657} } \\
    Used in section 649.
  \item \textbf{Determine vertical glue shrink setting, then \texttt{return or goto common\_ending 676} } \\
    Used in section 672.
  \item \textbf{Determine vertical glue stretch setting, then \texttt{return or goto common\_ending 673} } \\
    Used in section 672.
  \item \textbf{Discard erroneous prefixes and \texttt{return} 1212} \\
    Used in section 1211.
  \item \textbf{Discard the prefixes \texttt{\_long and \_outer} if they are irrelevant 1213} \\
    Used in section 1211.
  \item \textbf{Dispense with trivial cases of void or bad boxes 978} \\
    Used in section 977.
  \item \textbf{Display adjustment p 197} \\
    Used in section 183.
  \item \textbf{Display box p 184} \\
    Used in section 183.
  \item \textbf{Display choice node p 695} \\
    Used in section 690.
  \item \textbf{Display discretionary p 195} \\
    Used in section 183.
  \item \textbf{Display fraction node p 697} \\
    Used in section 690.
  \item \textbf{Display glue p 189} \\
    Used in section 183.
  \item \textbf{Display insertion p 188} \\
    Used in section 183.
\end{itemize}
\langle \text{Display kern } p \, 191 \rangle \quad \text{Used in section 183.}
\langle \text{Display leaders } p \, 190 \rangle \quad \text{Used in section 189.}
\langle \text{Display ligature } p \, 193 \rangle \quad \text{Used in section 183.}
\langle \text{Display mark } p \, 196 \rangle \quad \text{Used in section 183.}
\langle \text{Display math node } p \, 192 \rangle \quad \text{Used in section 183.}
\langle \text{Display node } p \, 183 \rangle \quad \text{Used in section 182.}
\langle \text{Display normal node } p \, 696 \rangle \quad \text{Used in section 690.}
\langle \text{Display penalty } p \, 194 \rangle \quad \text{Used in section 183.}
\langle \text{Display rule } p \, 187 \rangle \quad \text{Used in section 183.}
\langle \text{Display special fields of the unset node } p \, 185 \rangle \quad \text{Used in section 184.}
\langle \text{Display the current context } 312 \rangle \quad \text{Used in section 311.}
\langle \text{Display the insertion split cost } 1011 \rangle \quad \text{Used in section 1010.}
\langle \text{Display the page break cost } 1006 \rangle \quad \text{Used in section 1005.}
\langle \text{Display the token } (m,c) \, 294 \rangle \quad \text{Used in section 293.}
\langle \text{Display the value of } b \, 502 \rangle \quad \text{Used in section 498.}
\langle \text{Display the value of } glue_{set}(p) \, 180 \rangle \quad \text{Used in section 184.}
\langle \text{Display the whatis node } p \, 1356 \rangle \quad \text{Used in section 183.}
\langle \text{Display token } p, \text{ and return } \rangle \quad \text{if there are problems 293} \quad \text{Used in section 292.}
\langle \text{Do first-pass processing based on type } q; \text{ goto done_with_node } \rangle \quad \text{if a node has been fully processed, goto check_dimensions } \quad \text{if it has been translated into new_list } q, \text{ or goto done_with_node } \text{if a node has been fully processed } 728 \quad \text{Used in section 727.}
\langle \text{Do ligature or kern command, returning to main_fig_loop } \text{or main_loop_wrapup } \text{or main_loop_move } 1040 \rangle \quad \text{Used in section 1039.}
\langle \text{Do magic computation } 320 \rangle \quad \text{Used in section 292.}
\langle \text{Do some work that has been queued up for } \text{write } 1374 \rangle \quad \text{Used in section 1373.}
\langle \text{Drop current token and complain that it was unmatched } 1006 \rangle \quad \text{Used in section 1064.}
\langle \text{Dump a couple more things and the closing check word } 1320 \rangle \quad \text{Used in section 1302.}
\langle \text{Dump constants for consistency check } 1307 \rangle \quad \text{Used in section 1302.}
\langle \text{Dump regions } 1 \text{ to } 4 \text{ of eqtb } 1315 \rangle \quad \text{Used in section 1313.}
\langle \text{Dump regions } 5 \text{ and } 6 \text{ of eqtb } 1316 \rangle \quad \text{Used in section 1313.}
\langle \text{Dump the array info for internal font number } k \, 1322 \rangle \quad \text{Used in section 1320.}
\langle \text{Dump the dynamic memory } 1311 \rangle \quad \text{Used in section 1302.}
\langle \text{Dump the font information } 1320 \rangle \quad \text{Used in section 1302.}
\langle \text{Dump the hash table } 1318 \rangle \quad \text{Used in section 1313.}
\langle \text{Dump the hyphenation tables } 1324 \rangle \quad \text{Used in section 1302.}
\langle \text{Dump the string pool } 1309 \rangle \quad \text{Used in section 1302.}
\langle \text{Dump the table of equivalents } 1313 \rangle \quad \text{Used in section 1302.}
\langle \text{Either append the insertion node } p \text{ after node } q, \text{ and remove it from the current page, or delete node } (p \, 1022) \rangle \quad \text{Used in section 1020.}
\langle \text{Either insert the material specified by node } p \text{ into the appropriate box, or hold it for the next page; also delete node } p \text{ from the current page } 1020 \rangle \quad \text{Used in section 1014.}
\langle \text{Empty the last bytes out of } dvi_buf \, 509 \rangle \quad \text{Used in section 642.}
\langle \text{Ensure that box } 255 \text{ is empty after output } 1028 \rangle \quad \text{Used in section 1026.}
\langle \text{Ensure that box } 255 \text{ is empty before output } 1015 \rangle \quad \text{Used in section 1014.}
\langle \text{Ensure that } true_{max} \geq h + 256 \, 954 \rangle \quad \text{Used in section 953.}
\langle \text{Enter a hyphenation exception } 939 \rangle \quad \text{Used in section 935.}
\langle \text{Enter all of the patterns into a linked trie, until coming to a right brace } 961 \rangle \quad \text{Used in section 960.}
\langle \text{Enter as many hyphenation exceptions as are listed, until coming to a right brace; then return } 935 \rangle \quad \text{Used in section 934.}
\langle \text{Enter skip_blanks state, emit a space } 349 \rangle \quad \text{Used in section 347.}
Error handling procedures 78, 81, 82, 93, 94, 95, 1382* \(\) Used in section 4*.
Examine node \( p \) in the hlist, taking account of its effect on the dimensions of the new box, or moving it to the adjustment list; then advance \( p \) to the next node 651 \(\) Used in section 669.
Examine node \( p \) in the vlist, taking account of its effect on the dimensions of the new box; then advance \( p \) to the next node 669 \(\) Used in section 668.
Expand a nonmacro 367 \(\) Used in section 366.
Expand macros in the token list and make \( \text{link(defref)} \) point to the result 1371
Used in section 1370.
Expand the next part of the input 478 \(\) Used in section 477.
Expand the token after the next token 368 \(\) Used in section 367.
Explain that too many dead cycles have occurred in a row 1024 \(\) Used in section 1012.
Express astonishment that no number was here 446 \(\) Used in section 444.
Express consternation over the fact that no alignment is in progress 1128 \(\) Used in section 1127.
Express shock at the missing left brace; \( \text{goto found} \) 475 \(\) Used in section 474.
Feed the macro body and its parameters to the scanner 390 \(\) Used in section 389.
Fetch a box dimension 430 \(\) Used in section 413.
Fetch a character code from some table 414 \(\) Used in section 413.
Fetch a font dimension 425 \(\) Used in section 413.
Fetch a font integer 426 \(\) Used in section 413.
Fetch a register 427 \(\) Used in section 413.
Fetch a token list or font identifier, provided that \( \text{level = tok_val} \) 415 \(\) Used in section 413.
Fetch an internal dimension and \( \text{goto attach_sign} \), or fetch an internal integer 449
Used in section 448.
Fetch an item in the current node, if appropriate 424 \(\) Used in section 413.
Fetch something on the \( \text{page_so_far} \) 421 \(\) Used in section 413.
Fetch the \( \text{dead_cycles} \) or the \( \text{insert_penalties} \) 419 \(\) Used in section 413.
Fetch the \( \text{par_shape} \) size 423 \(\) Used in section 413.
Fetch the \( \text{prev_graf} \) 422 \(\) Used in section 413.
Fetch the \( \text{space_factor} \) or the \( \text{prev_depth} \) 418 \(\) Used in section 413.
Find an active node with fewest demerits 874 \(\) Used in section 873.
Find hyphen locations for the word in \( \text{hc} \), or return 923 \(\) Used in section 895.
Find optimal breakpoints 863 \(\) Used in section 845.
Find the best active node for the desired looseness 875 \(\) Used in section 873.
Find the best way to split the insertion, and change \( \text{type(r)} \) to \( \text{split_up} \) 1010
Used in section 1008.
Find the glue specification, \( \text{main_p} \), for text spaces in the current font 1012
Used in sections 1011 and 1043.
Finish an alignment in a display 1206 \(\) Used in section 812.
Finish displayed math 1199 \(\) Used in section 1194.
Finish issuing a diagnostic message for an overfull or underfull hbox 663 \(\) Used in section 649.
Finish issuing a diagnostic message for an overfull or underfull vbox 675 \(\) Used in section 668.
Finish line, emit a \( \text{\textbackslash par} \) 351 \(\) Used in section 347.
Finish line, emit a space 348 \(\) Used in section 347.
Finish line, \( \text{goto switch} \) 350 \(\) Used in section 347.
Finish math in text 1196 \(\) Used in section 1194.
Finish the DVI file 642 \(\) Used in section 1333*.
Finish the extensions 1378 \(\) Used in section 1333*.
Fire up the user’s output routine and return 1025 \(\) Used in section 1012.
Fix the reference count, if any, and negate \( \text{cur_val} \) if \( \text{negative} \) 430 \(\) Used in section 433.
Flush the box from memory, showing statistics if requested 639 \(\) Used in section 638.
Forbidden cases detected in \( \text{main_control} \) 1048, 1098, 1111, 1144 \(\) Used in section 1045.
Generate a \( \text{down} \) or \( \text{right} \) command for \( w \) and return 610 \(\) Used in section 607.
(Generate a $\varnothing$ or $\emptyset$ command in order to reuse a previous appearance of $w$ 609)

Used in section 607.

(Generate ready to compress the trie 952) Used in section 906.

(Generate ready to start line breaking 816*, 827, 834, 848) Used in section 815.

(Get the first line of input and prepare to start 1337) Used in section 1332*.

(Get the next non-blank non-call token 406) Used in sections 405, 441, 455, 503, 526, 577, 785, 791, and 1045.

(Get the next non-blank non-relax non-call token 404) Used in sections 403, 1078, 1084, 1151, 1160, 1211, 1226, and 1270.

(Get the next non-blank non-sign token; set $negative$ appropriately 441) Used in sections 440, 448, and 461.

(Get the next token, suppressing expansion 358) Used in section 357.

(Get user's advice and return 83) Used in section 82.

(Give diagnostic information, if requested 1031) Used in section 1030.

(Give incorrect hyphenation error 996) Used in section 935.


(Generate display math mode 1145) Used in section 1138.

(Generate ordinary math mode 1139) Used in sections 1138 and 1142.

(Go through the preamble list, determining the column widths and changing the alignrecords to dummy unset boxes 801) Used in section 800.

(Grow more variable-size memory and goto restart 126) Used in section 125.

(Handle situations involving spaces, braces, changes of state 347) Used in section 344.

(If a line number class has ended, create new active nodes for the best feasible breaks in that class; then return if $r = last active$, otherwise compute the new line_width 835) Used in section 829.

(If all characters of the family fit relative to $h$, then goto found, otherwise goto not_found 955) Used in section 953.

(If an alignment entry has just ended, take appropriate action 342) Used in section 341.

(If an expanded code is present, reduce it and goto start cs 355) Used in sections 354 and 356.

(If dumping is not allowed, abort 1304) Used in section 1302.

(If instruction cur_j is a kern with cur_c, attach the kern after q; or if it is a ligature with cur_c, combine noads q and p appropriately; then return if the cursor has moved past a noad, or goto restart 733) Used in section 752.

(If no hyphens were found, return 902) Used in section 895.

(If node cur_p is a legal breakpoint, call try_break; then update the active widths by including the glue in glue_ptr (cur_p) 868) Used in section 866.

(If node p is a legal breakpoint, check if this break is the best known, and goto done if p is null or if the page so far is already too full to accept more stuff 972) Used in section 970.

(If node q is a style node, change the style and goto delete q; otherwise if it is not a noad, put it into the hlist, advance q, and goto done; otherwise set s to the size of noad q, set t to the associated type (ord_noad .. inner_noad), and set pen to the associated penalty 761) Used in section 760.

(If node r is of type delta_node, update cur_active_width, set prev_r and prev_prev_r, then goto continue 832) Used in section 829.

(If the current list ends with a box node, delete it from the list and make cur_box point to it; otherwise set cur_box <- null 1080) Used in section 1079.

(If the current page is empty and node p is to be deleted, goto done1; otherwise use node p to update the state of the current page; if this node is an insertion, goto contribute; otherwise
if this node is not a legal breakpoint, \texttt{goto contribute} or \texttt{update_heights}; otherwise set \texttt{pi} to the penalty associated with this breakpoint 1000 \texttt{) Used in section 997.}

\langle If the cursor is immediately followed by the right boundary, \texttt{goto reswitch}; if it’s followed by an invalid character, \texttt{goto big\_switch}; otherwise move the cursor one step to the right and \texttt{goto main\_big\_loop} 1036 \texttt{) Used in section 1034.}

\langle If the next character is a parameter number, make \texttt{cur\_fok} a \texttt{match} token; but if it is a left brace, store \texttt{left\_brace, end\_match}, set \texttt{hash\_brace}, and \texttt{goto done} 476 \texttt{) Used in section 474.}

\langle If the preamble list has been traversed, check that the row has ended 792 \texttt{) Used in section 791.}

\langle If the right-hand side is a token parameter or token register, finish the assignment and \texttt{goto done} 1227 \texttt{) Used in section 1226.}

\langle If the string \texttt{hyph\_word[h]} is less than \texttt{hc[1..hn]}, \texttt{goto not\_found}; but if the two strings are equal, set \texttt{hys} to the hyphen positions and \texttt{goto found} 931 \texttt{) Used in section 930.}

\langle If the string \texttt{hyph\_word[h]} is less than or equal to \texttt{s}, interchange \langle \texttt{hyph\_word[h], hyph\_list[h]} \rangle with \langle \texttt{s, p} \rangle 941 \texttt{) Used in section 940.}

\langle If there’s a ligature or kern at the cursor position, update the data structures, possibly advancing \texttt{j}; continue until the cursor moves 909 \texttt{) Used in section 906.}

\langle If there’s a ligature/kern command relevant to \texttt{cur\_f} and \texttt{cur\_r}, adjust the text appropriately; exit to \textbf{main\_loop\_wrapup} 1030 \texttt{) Used in section 1034.}

\langle If this font has already been loaded, set \texttt{f} to the internal font number and \texttt{goto common\_ending} 1260 \texttt{) Used in section 1257.}

\langle If this \texttt{sup\_mark} starts an expanded character like \^\^A or \^\^df, then \texttt{goto reswitch}, otherwise set \texttt{state ← mid\_line 352} \texttt{) Used in section 344.}

\langle Ignore the fraction operation and complain about this ambiguous case 1183 \texttt{) Used in section 1181.}

\langle Implement \texttt{closeout} 1353 \texttt{) Used in section 1348.}

\langle Implement \texttt{immediate} 1375 \texttt{) Used in section 1348.}

\langle Implement \texttt{openout} 1351 \texttt{) Used in section 1348.}

\langle Implement \texttt{setlanguage} 1377 \texttt{) Used in section 1348.}

\langle Implement \texttt{special} 1354 \texttt{) Used in section 1348.}

\langle Implement \texttt{write} 1352 \texttt{) Used in section 1348.}

\langle Incorporate a whatsit node into a \texttt{vbox} 1359 \texttt{) Used in section 669.}

\langle Incorporate a whatsit node into an \texttt{bbox} 1360 \texttt{) Used in section 651.}

\langle Incorporate box dimensions into the dimensions of the \texttt{bbox} that will contain it 653 \texttt{) Used in section 651.}

\langle Incorporate box dimensions into the dimensions of the \texttt{vbox} that will contain it 670 \texttt{) Used in section 669.}

\langle Incorporate character dimensions into the dimensions of the \texttt{bbox} that will contain it, then move to the next node 654 \texttt{) Used in section 651.}

\langle Incorporate glue into the horizontal totals 656 \texttt{) Used in section 651.}

\langle Incorporate glue into the vertical totals 671 \texttt{) Used in section 669.}

\langle Increase the number of parameters in the last \texttt{font} 580 \texttt{) Used in section 578.}

\langle Initialize for hyphenating a paragraph 891 \texttt{) Used in section 863.}

\langle Initialize table entries (done by \texttt{INITEX only}) 164, 222, 228, 232, 240, 250, 258, 552, 946, 951, 1216, 1301, 1360 \texttt{) Used in section 8.}

\langle Initialize the current page, insert the \texttt{\topskip} glue ahead of \texttt{p}, and \texttt{goto continue} 1001 \texttt{) Used in section 1000.}

\langle Initialize the input routines 331 \texttt{) Used in section 1337.}

\langle Initialize the output routines 55, 61, 528, 533 \texttt{) Used in section 1332*.}

\langle Initialize the print \texttt{selector} based on \texttt{interaction} 75 \texttt{) Used in sections 1265* and 1337.}

\langle Initialize the special list selector and constant nodes 790, 797, 820, 981, 988 \texttt{) Used in section 164.}

\langle Initialize variables as \texttt{skip\_out} begins 617 \texttt{) Used in section 640.}

\langle Initialize whatever \TeX\ might access 8 \texttt{) Used in section 4*.}
Let \( h \) insert a new pattern into the linked trie.

Lengthen the preamble periodically.

Insert glue for \( i \).

Insert a new node between \( q \) and \( p \), and make \( p \) point to it.

Initiate the construction of an hbox or vbox, then \textbf{return} 1083.

Input and store tokens from the next line of the file 483.

Input for \texttt{read} from the terminal 484.

Input from external file, \texttt{goto restart} if no input found 343.

Input from token list, \texttt{goto restart} if end of list or if a parameter needs to be expanded 357.

Used in section 341.

Input the first line of \texttt{read_file}[m] 485.

Input the next line of \texttt{read_file}[m] 486.

Insert a delta node to prepare for breaks at cur_p 843.

Insert a delta node to prepare for the next active node 844.

Insert a dummy node to be sub/superscripted 1177.

Insert a new active node from \texttt{best_place} to \texttt{cur_p} 845.

Insert a new control sequence after \( p \), then make \( p \) point to it 260.

Insert a new pattern into the linked trie 963.

Insert a new trie node between \( q \) and \( p \), and make \( p \) point to it 964.

Insert a token containing \texttt{frozen_addr} 375.

Insert a token saved by \texttt{afterassignment}, if any 1269.

Insert glue for \texttt{split_top_skip} and set \( p \leftarrow \texttt{null} \) 969.

Insert hyphens as specified in \texttt{hyph_list}[h] 932.

Insert macro parameter and \texttt{goto restart} 379.

Insert the appropriate mark text into the scanner 386.

Insert the current list into its environment 812.

Insert the pair \((s,p)\) into the exception table 940.

Insert the \((v_j)\) template and \texttt{goto restart} 789.

Insert token \( p \) into \TeX{}'s input 326.

Interpret code \( c \) and \texttt{return} if done 84.

Introduce new material from the terminal and \texttt{return} 87.

Issue an error message if cur_val = \texttt{fnem_ptr} 579.

Justify the line ending at breakpoint \texttt{cur_p}, and append it to the current vertical list, together with associated penalties and other insertions 880.

Labels in the outer block \( 6 \)  6.

Last-minute procedures 1333, 1335, 1336, 1338, 1380.

Lengthen the preamble periodically 793.

Let \texttt{cur_h} be the position of the first box, and set \texttt{leader_red} + \( l_x \) to the spacing between corresponding parts of boxes 627.

Let \texttt{cur_v} be the position of the first box, and set \texttt{leader_ht} + \( l_x \) to the spacing between corresponding parts of boxes 636.

Let \( d \) be the natural width of node \( p \); if the node is “visible,” \texttt{goto found}; if the node is glue that stretches or shrinks, set \( v \leftarrow \texttt{max_dimen} \) 1147.

Let \( d \) be the natural width of this glue; if stretching or shrinking, set \( v \leftarrow \texttt{max_dimen} \), \texttt{goto found} in the case of leaders 1148.

Let \( d \) be the width of the whatsis \( p \) 1361.

Let \( n \) be the largest legal code value, based on \texttt{cur_chr} 1233.

Link node \( p \) into the current page and \texttt{goto done} 998.

Local variables for dimension calculations 450.

Local variables for finishing a displayed formula 1198.

Local variables for formatting calculations 315.

Local variables for hyphenation 901, 912, 922, 929.

Local variables for initialization 19, 163, 927.

Local variables for line breaking 862, 893.
Look ahead for another character, or leave lig_stack empty if there's none there 1038
Used in section 1034.

Look at all the marks in nodes before the break, and set the final link to null at the break 979
Used in section 977.

Look at the list of characters starting with x in font g; set f and c whenever a better character is found; goto found as soon as a large enough variant is encountered 708
Used in section 707.

Look at the other stack entries until deciding what sort of DVI command to generate; goto found if node p is a "hit" 611
Used in section 607.

Look at the variants of (x, y); set f and c whenever a better character is found; goto found as soon as a large enough variant is encountered 707
Used in section 706.

Look for parameter number or ## 479
Used in section 477.

Look for the word hc[l . . . hn] in the exception table, and goto found (with hyf containing the hyphens) if an entry is found 930
Used in section 923.

Look up the characters of list r in the hash table, and set cur_r 374
Used in section 372.

Make a copy of node p in node r 205
Used in section 204.

Make a ligature node, if ligature_present; insert a null discretionary, if appropriate 1035
Used in section 1034.

Make a partial copy of the whatsit node p and make r point to it; set words to the number of initial words not yet copied 1357
Used in section 206.

Make a second pass over the nlist, removing all noods and inserting the proper spacing and penalties 760
Used in section 726.

Make final adjustments and goto done 576
Used in section 562.

Make node p look like a char_node and goto reswitch 652
Used in sections 622, 655, and 1147.

Make sure that page_max_depth is not exceeded 1003
Used in section 997.

Make sure that pi is in the proper range 831
Used in section 829.

Make the contribution list empty by setting its tail to contrib_head 965
Used in section 994.

Make the first 256 strings 48
Used in section 47.

Make the height of box y equal to h 739
Used in section 738.

Make the running dimensions in rule q extend to the boundaries of the alignment 806
Used in section 805.

Make the unset node r into a vlist_node of height w; setting the glue as if the height were t 811
Used in section 808.

Make the unset node r into an hlist_node of width w, setting the glue as if the width were t 810
Used in section 808.

Make variable b point to a box for (f, c) 710
Used in section 706.

Manufacture a control sequence name 372
Used in section 367.

Math-only cases in non-math modes, or vice versa 1016
Used in section 1045.

Merge the widths in the span nodes of q with those of p, destroying the span nodes of q 803
Used in section 801.

Modify the end of the line to reflect the nature of the break and to include \rightskip; also set the proper value of disc_break 881
Used in section 880.

Modify the glue specification in main_p according to the space factor 1044
Used in section 1043.

Move down or put out leaders 634
Used in section 631.

Move node p to the current page; if it is time for a page break, put the nodes following the break back onto the contribution list, and return to the user's output routine if there is one 997
Used in section 994.

Move pointer s to the end of the current list, and set replace_count(r) appropriately 918
Used in section 914.

Move right or output leaders 625
Used in section 622.

Move the characters of a ligature node to hu and hc; but goto done3 if they are not all letters 898
Used in section 897.
\begin{verbatim}
\begin{enumerate}
  \item Move the cursor past a pseudo-ligature, then \texttt{goto main_loop lookahead} or \texttt{main_loop} 1037 \newline
    Used in section 1034.
  \item Move the data into \texttt{trie} 958 \newline
    Used in section 966.
  \item Move to next line of file, or \texttt{goto restart} if there is no next line, or \texttt{return} if a \texttt{read} line has \newline
    finished 360* \newline
    Used in section 343.
  \item Negate all three glue components of \texttt{cur_val} 431 \newline
    Used in section 430.
  \item Nullify \texttt{width(q)} and the tabskip glue following this column 802 \newline
    Used in section 801.
  \item Numbered cases for \texttt{debug_help} 1339* \newline
    Used in section 1338*.
  \item Open \texttt{tm_file} for input 563 \newline
    Used in section 562.
  \item Other local variables for \texttt{try_break} 830 \newline
    Used in section 829.
  \item Output a box in a vlist 632 \newline
    Used in section 631.
  \item Output a box in an hlist 623 \newline
    Used in section 622.
  \item Output a leader box at \texttt{cur_h}, then advance \texttt{cur_h} by \texttt{leader wd + lx} 628 \newline
    Used in section 626.
  \item Output a leader box at \texttt{cur_v}, then advance \texttt{cur_v} by \texttt{leader ht + lx} 637 \newline
    Used in section 635.
  \item Output a rule in a vlist, \texttt{goto next p} 633 \newline
    Used in section 631.
  \item Output a rule in an hlist 624 \newline
    Used in section 622.
  \item Output leaders in a vlist, \texttt{goto fin rule} if a rule or to \texttt{next p} if done 635 \newline
    Used in section 634.
  \item Output leaders in an hlist, \texttt{goto fin rule} if a rule or to \texttt{next p} if done 626 \newline
    Used in section 625.
  \item Output node \texttt{p} for \texttt{hlist out} and move to the next node, maintaining the condition \newline
    \texttt{cur v = base line} 620 \newline
    Used in section 619.
  \item Output node \texttt{p} for \texttt{vlist out} and move to the next node, maintaining the condition \newline
    \texttt{cur h = left edge} 630 \newline
    Used in section 629.
  \item Output statistics about this job 1334 \newline
    Used in section 1333*.
  \item Output the font definitions for all fonts that were used 643 \newline
    Used in section 642.
  \item Output the font name whose internal number is \texttt{f} 603 \newline
    Used in section 602.
  \item Output the non-char_node \texttt{p} for \texttt{hlist out} and move to the next node 622 \newline
    Used in section 620.
  \item Output the non-char_node \texttt{p} for \texttt{vlist out} 631 \newline
    Used in section 630.
  \item Output the what sit \texttt{p} in a vlist 1306 \newline
    Used in section 631.
  \item Output the what sit \texttt{p} in an hlist 1307 \newline
    Used in section 622.
  \item Pack the family into \texttt{trie} relative to \texttt{h} 956 \newline
    Used in section 953.
  \item Package an unset box for the current column and record its width 796 \newline
    Used in section 791.
  \item Package the preamble list, to determine the actual tabskip glue amounts, and let \texttt{p} point to this \newline
    prototype box 801 \newline
    Used in section 800.
  \item Perform the default output routine 1023 \newline
    Used in section 1012.
  \item Pontificate about improper alignment in display 1207 \newline
    Used in section 1206.
  \item Pop the condition stack 496 \newline
    Used in sections 498, 500, 509, and 510.
  \item Prepare all the boxes involved in insertions to act as queues 1018 \newline
    Used in section 1014.
  \item Prepare to deactivate node \texttt{r}, and \texttt{goto deactivate} unless there is a reason to consider lines of \newline
    text from \texttt{r} to \texttt{cur p} 854 \newline
    Used in section 851.
  \item Prepare to insert a token that matches \texttt{cur group}, and print what it is 1065 \newline
    Used in section 1061.
  \item Prepare to move a box or rule node to the current page, then \texttt{goto contribute} 1002 \newline
    Used in section 1000.
  \item Prepare to move what sit \texttt{p} to the current page, then \texttt{goto contribute} 1364 \newline
    Used in section 1000.
  \item Print a short indication of the contents of node \texttt{p} 175 \newline
    Used in section 174.
  \item Print a symbolic description of the new break node 846 \newline
    Used in section 845.
  \item Print a symbolic description of this feasible break 856 \newline
    Used in section 855.
  \item Print either `definition' or `use' or `preamble' or `text', and insert tokens that should lead \newline
    to recovery 330 \newline
    Used in section 338.
  \item Print location of current line 313 \newline
    Used in section 312.
  \item Print newly busy locations 171 \newline
    Used in section 167.
  \item Print string \texttt{s} as an error message 1283 \newline
    Used in section 1279.
\end{enumerate}
\end{verbatim}
(Print string \texttt{s} on the terminal \texttt{1280})  Used in section \texttt{1279}.

(Print the banner line, including the date and time \texttt{536})  Used in section \texttt{534}.

(Print the help identifier for \texttt{font(p)} \texttt{267})  Used in sections \texttt{174} and \texttt{176}.

(Print the help information and \texttt{goto continue 89})  Used in section \texttt{84*}.

(Print the list between \texttt{printed_node} and \texttt{cur_p}, then set \texttt{printed_node} \texttt{\leftarrow cur_p} \texttt{857})

Used in section \texttt{856}.

(Print the menu of available options \texttt{85})  Used in section \texttt{84*}.

(Print the result of command \texttt{e} \texttt{472})  Used in section \texttt{470}.

(Print two lines using the tricky pseudoprinted information \texttt{317})  Used in section \texttt{312}.

(Print type of token list \texttt{314})  Used in section \texttt{312}.

(Process an active-character control sequence and set \texttt{state \leftarrow midline 353})  Used in section \texttt{344}.

(Process node-or-node \texttt{q} as much as possible in preparation for the second pass of \texttt{mlist_to_hlist},

then move to the next item in the mlist \texttt{727})  Used in section \texttt{726}.

(Process whatsit \texttt{p} in \texttt{vertbreak} loop, \texttt{goto not_found} \texttt{1365})  Used in section \texttt{973}.

(Prune the current list, if necessary, until it contains only \texttt{char_node, kern_node, hlist_node, vlist_node, rule_node, and ligature_node} items; set \texttt{n} to the length of the list, and set \texttt{q} to the list's tail \texttt{1121})  Used in section \texttt{1119}.

(Prune unwanted nodes at the beginning of the next line \texttt{879*})  Used in section \texttt{877*}.

(Pseudoprint the line \texttt{318})  Used in section \texttt{312}.

(Pseudoprint the token list \texttt{319})  Used in section \texttt{312}.

(Push the condition stack \texttt{495})  Used in section \texttt{498}.

(Put each of \TeX's primitives into the hash table \texttt{236, 230, 238, 248, 265, 334, 376, 384, 411, 416, 468, 487, 491, 553, 780, 983, 1052, 1058, 1071, 1088, 1107, 1114, 1141, 1145, 1156, 1169, 1178, 1188, 1208, 1219, 1222, 1230, 1250, 1254, 1362, 1277, 1280, 1291, 1344})  Used in section \texttt{1336}.

(Put help message on the transcript file \texttt{90})  Used in section \texttt{82}.

(Put the characters \texttt{hu \mid i + 1 ..} into \texttt{postbreak (r)}, appending to this list and to \texttt{major_tail} until synchronization has been achieved \texttt{916})  Used in section \texttt{914}.

(Put the characters \texttt{hu /l . . i]} and a hyphen into \texttt{pre_break (r)} \texttt{915})  Used in section \texttt{914}.

(Put the fraction into a box with its delimiters, and make \texttt{new_hlist (q) point to it} \texttt{748})

Used in section \texttt{743}.

(Put the \texttt{\leftskip} glue at the left and detach this line \texttt{887})  Used in section \texttt{880}.

(Put the optimal current page into box \texttt{255}, update \texttt{first_mark} and \texttt{bot_mark}, append insertions to their boxes, and put the remaining nodes back on the contribution list \texttt{1014})

Used in section \texttt{1012}.

(Put the (positive) 'at' size into \texttt{s} \texttt{1250})  Used in section \texttt{1258}.

(Put the \texttt{\rightskip} glue after node \texttt{q} \texttt{886})  Used in section \texttt{881}.

(Read and check the font data; \texttt{abort} if the \texttt{TFM} file is malformed; if there's no room for this font, say so and \texttt{goto done}; otherwise \texttt{incr (font_ptr)} and \texttt{goto done} \texttt{562})  Used in section \texttt{560}.

(Read box dimensions \texttt{571})  Used in section \texttt{562}.

(Read character data \texttt{569})  Used in section \texttt{562}.

(Read extensible character recipes \texttt{574})  Used in section \texttt{562}.

(Read font parameters \texttt{575*})  Used in section \texttt{562}.

(Read ligature/kern program \texttt{573})  Used in section \texttt{562}.

(Read next line of file into \texttt{buffer}, or \texttt{goto restart} if the file has ended \texttt{362})  Used in section \texttt{360*}.

(Read one string, but return \texttt{false} if the string memory space is getting too tight for comfort \texttt{52})

Used in section \texttt{51*}.

(Read the first line of the new file \texttt{538})  Used in section \texttt{537*}.

(Read the other strings from the \texttt{TEX_POOL} file and return \texttt{true}, or give an error message and return \texttt{false} \texttt{51*})  Used in section \texttt{47}.

(Read the \texttt{TFM} header \texttt{568})  Used in section \texttt{562}.

(Read the \texttt{TFM} size fields \texttt{565})  Used in section \texttt{562}.

(Readjust the height and depth of \texttt{cur_box}, for \texttt{vtop} \texttt{1087})  Used in section \texttt{1086}.
(Reconstitute nodes for the hyphenated word, inserting discretionary hyphens 913)  
Used in section 903.
(Record a new feasible break 855)  Used in section 851.
(Recover from an unbalanced output routine 1027)  Used in section 1026.
(Recover from an unbalanced write command 1372)  Used in section 1371.
(Recycle node p 999)  Used in section 997.
(Remove the last box, unless it’s part of a discretionary 1081)  Used in section 1080.
(Replace nodes ha . . . hb by a sequence of nodes that includes the discretionary hyphens 903)  
Used in section 895.
(Replace the tail of the list by p 1187)  Used in section 1186.
(Replace z by z′ and compute α, β 572)  Used in section 571.
(Report a runaway argument and abort 396)  Used in sections 392 and 399.
(Report a tight hbox and goto common ending, if this box is sufficiently bad 667)  
Used in section 664.
(Report a tight vbox and goto common ending, if this box is sufficiently bad 678)  
Used in section 676.
(Report an extra right brace and goto continue 305)  Used in section 302.
(Report an improper use of the macro and abort 390)  Used in section 397.
(Report an overfull hbox and goto common ending, if this box is sufficiently bad 666)  
Used in section 664.
(Report an overfull vbox and goto common ending, if this box is sufficiently bad 676)  
Used in section 676.
(Report an underfull hbox and goto common ending, if this box is sufficiently bad 660)  
Used in section 668.
(Report an underfull vbox and goto common ending, if this box is sufficiently bad 674)  
Used in section 673.
(Report overflow of the input buffer, and abort 35)  Used in sections 31* and 36*.
(Report that an invalid delimiter code is being changed to null; set cur val ← 0 1161)  
Used in section 1160.
(Report that the font won’t be loaded 561)  Used in section 560.
(Report that this dimension is out of range 460)  Used in section 448.
(Resume the page builder after an output routine has come to an end 1026)  Used in section 1100.
(Reverse the links of the relevant passive nodes, setting cur p to the first breakpoint 878)  
Used in section 877*.
(Scan a control sequence and set state ← skip blanks or mid line 354)  Used in section 344.
(Scan a numeric constant 444)  Used in section 440.
(Scan a parameter until its delimiter string has been found; or, if s = null, simply scan the  
delimiter string 392)  Used in section 391.
(Scan a subformula enclosed in braces and return 1153)  Used in section 1151.
(Scan ahead in the buffer until finding a nonletter; if an expanded code is encountered, reduce  
it and goto start cs; otherwise if a multiletter control sequence is found, adjust cur cs and  
loc, and goto found 356)  Used in section 354.
(Scan an alphabetic character code into cur val 442)  Used in section 440.
(Scan an optional space 443)  Used in sections 442, 448, 455, and 1200.
(Scan and build the body of the token list; goto found when finished 477)  Used in section 473.
(Scan and build the parameter part of the macro definition 474)  Used in section 473.
(Scan decimal fraction 452)  Used in section 448.
(Scan file name in the buffer 531)  Used in section 530*.
(Scan for all other units and adjust cur val and f accordingly; goto done in the case of scaled  
points 458)  Used in section 453.
(Scan for fill units; goto attach fraction if found 454)  Used in section 453.
(Scan for mu units and goto attach fraction 456)  Used in section 453.
(Scan for units that are internal dimensions; \texttt{goto attach\_sign} with \texttt{cur\_val} set if found 455)

Used in section 453.

(Scan preamble text until \texttt{cur\_cmd} is \texttt{tab\_mark} or \texttt{cur\_ret}, looking for changes in the tabskip glue; append an alignrecord to the preamble list 779) Used in section 777.

(Scan the argument for command \texttt{c} 471) Used in section 470.

(Scan the font size specification 1258) Used in section 1257.

(Scan the parameters and make \texttt{link}(r) point to the macro body; but \texttt{return} if an illegal \texttt{\par} is detected 391) Used in section 389.

(Scan the preamble and record it in the \texttt{preamble} list 777) Used in section 774.

(Scan the template \langle \texttt{uj} \rangle, putting the resulting token list in \texttt{hold\_head} 783) Used in section 779.

(Scan the template \langle \texttt{uj} \rangle, putting the resulting token list in \texttt{hold\_head} 784) Used in section 779.

(Scan units and set \texttt{cur\_val} to \( x \cdot (\texttt{cur\_val} + f/2^{16}) \), where there are \( x \) sp per unit; \texttt{goto attach\_sign} if the units are internal 453) Used in section 448.

(Scan \texttt{eqtb} for equivalents equal to \( p \) 255) Used in section 172.

(Scan \texttt{hyph\_list} for pointers to \( p \) 933) Used in section 172.

(Scan \texttt{save\_stack} for equivalents that point to \( p \) 285) Used in section 172.

(Select the appropriate case and \texttt{return} or \texttt{goto common\_ending} 569) Used in section 501.

(Set initial values of key variables 21, 23*, 24, 74, 77, 80*, 97, 106, 215, 254, 257, 272, 287, 383, 439, 481, 490, 521*, 551, 556, 593, 596, 606, 648, 662, 685, 771, 928, 990, 1033, 1267, 1282, 1300, 1343, 1381*)

Used in section 8.

(Set line length parameters in preparation for hanging indentation 849) Used in section 848.

(Set the glue in all the unset boxes of the current list 805) Used in section 800.

(Select the appropriate case and \texttt{return} or \texttt{goto common\_ending} 569) Used in section 501.

(Select the appropriate case and \texttt{return} or \texttt{goto common\_ending} 569) Used in section 501.

(Select the some parameter or register, then \texttt{goto common\_ending} 1297) Used in section 1293.

(Show the current value of some parameter or register, then \texttt{goto common\_ending} 1294) Used in section 1293.

(Show the macro being expanded 401) Used in section 389.

(Show the page in the current contents of a box 1296) Used in section 1293.

(Show the current meaning of a token, then \texttt{goto common\_ending} 1294) Used in section 1293.

(Show the current value of some parameter or register, then \texttt{goto common\_ending} 1297) Used in section 1293.

(Show the font identifier in \texttt{eqtb}[n] 234) Used in section 233.

(Show the halfword code in \texttt{eqtb}[n] 235) Used in section 233.

(Show the status of the current page 986) Used in section 218.

(Show the text of the macro being expanded 401) Used in section 389.

(Show the auxiliary field, \texttt{f} 219) Used in section 218.

(Show the current contents of a box 1296) Used in section 1293.

(Show the current meaning of a token, then \texttt{goto common\_ending} 1294) Used in section 1293.

(Show the current value of some parameter or register, then \texttt{goto common\_ending} 1297) Used in section 1293.

(Show the font identifier in \texttt{eqtb}[n] 234) Used in section 233.

(Show the halfword code in \texttt{eqtb}[n] 235) Used in section 233.

(Show the status of the current page 986) Used in section 218.

(Show the text of the macro being expanded 401) Used in section 389.

(Show a trivial box 721) Used in section 720.

(Show a trivial box 721) Used in section 720.

(Skip to \texttt{\else} or \texttt{\fi}, then \texttt{goto common\_ending} 500) Used in section 408.

(Skip to node \texttt{ha}, or \texttt{goto done1} if no hyphenation should be attempted 896) Used in section 894.
\begin{itemize}
\item Update the active widths, since the first active node has been deleted. \hspace{1em} Used in section 860.
\item Update the current height and depth measurements with respect to a glue or kern node \( p \). \hspace{1em} Used in section 976.
\item Update the current page measurements with respect to the glue or kern specified by node \( p \). \hspace{1em} Used in section 997.
\item Update the value of \texttt{printed\_node} for symbolic displays. \hspace{1em} Used in section 829.
\item Update the values of \texttt{first\_mark} and \texttt{bot\_mark}. \hspace{1em} Used in section 1014.
\item Update the values of \texttt{last\_glue}, \texttt{last\_penalty}, and \texttt{last\_kern}. \hspace{1em} Used in section 994.
\item Update the values of \texttt{max\_h} and \texttt{max\_v}; but if the page is too large, \texttt{goto done}. \hspace{1em} Used in section 641.
\item Update width entry for spanned columns. \hspace{1em} Used in section 796.
\item Use code \( c \) to distinguish between generalized fractions. \hspace{1em} Used in section 1182.
\item Use node \( p \) to update the current height and depth measurements; if this node is not a legal breakpoint, \texttt{goto not\_found} or \texttt{update\_heights}, otherwise set \( pi \) to the associated penalty at the break. \hspace{1em} Used in section 972.
\item Use size fields to allocate font information. \hspace{1em} Used in section 566.
\item Wipe out the whatsit node \( p \) and \texttt{goto done}. \hspace{1em} Used in section 202.
\item Wrap up the box specified by node \( r \), splitting node \( p \) if called for; set \( wait \leftarrow true \) if node \( p \) holds a remainder after splitting. \hspace{1em} Used in section 1020.
\end{itemize}