0.* About TEX$_{GPC}$. TEX$_{GPC}$ is a Unix implementation of Donald E. Knuth's TEX82 in the version 3.14159265 from January 2014. It is based on GNU Pascal. The accompaning `README` file tells you how to build and run TEX$_{GPC}$. To help you identify the differences of TEX82 and TEX$_{GPC}$, 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 TEX82
e fixes a small error in TEX82
F adds a feature as suggested by Knuth
P removes a violation of Pascal (Jensen, Wirth: *Pascal User Manual and Report*, 3rd edition, 1985)
G a GNU Pascal extension (Version 20070904)
U make Unix happy
u make Unix user happy
h make Helbig happy
N a note that helped me to understand this program.
B a bug I couldn't fix.

Identifiers that come with GNU Pascal are coded as `WEB` macros and prefixed by '*gpc_*'. That helps to resolve name clashes.

TEX$_{GPC}$ is slightly slower than web2c based programs. To compile the device independend file for this document, the web2c version from TEX Life 2008 ran 1.1 seconds and TEX$_{GPC}$ 1.2 seconds.

Going with Dijkstra, see `http://www.cs.utexas.edu/users/EWD/videos/noorderlicht.mpg`, I don't believe in version numbers, since I don't believe in maintaining software—I consider TEX$_{GPC}$ finished—and it must go without a number. This does not mean that I don't care any more about TEX$_{GPC}$; comments or questions are quite welcome. In fact, I tried to explain why I changed what and how in order to encourage you to undertake further modifications or bugfixes yourself and I'll be glad to help.

I wish to thank Frank Heckenbach and Emil Jerabek from the GNU Pascal mailing list for clarifying GPC's I/O buffering strategies, and David Kastrup from the `de.comp.text.tex` news group for enlightening articles on some of TEX's more obscure features and for discussing the 'empty last line error'.

The 2008 edition of TEX$_{GPC}$ was tested on NetBSD 3.1 and GPC 20020510, which happens to be the version of GNU Pascal offered in the NetBSD package collection. This edition was run on Mac OS X 10.6.2 and GPC 20070904. Most people seem to run TEX$_{GPC}$ with the current GPC version, which caused trouble in two cases: (1) The integer parameter to be passed to *gpc_install_signal_handler* is now of type *cinteger* instead of *integer*. (2) The function *gpc_install_signal_handler* was mistreated as a procedure by TEX$_{GPC}$, and the current version of GPC won't let you go away with that anymore. Luis Rivera and Martin Monperrus detected this error, and Martin suggested to publish an edition that runs with the current version of GPC which is this one.

Joachim Kuebart spotted another error in the 2008 edition. When the first line consists of blanks only, a loop is not terminated properly. Joachim found this by reading this document, not by running TEX$_{GPC}$. This in turn motivated me to improve the comments resulting in a lot of changes.

TEX$_{GPC}$ slightly differs from TEX: On input, trailing blanks are not removed, on input of the first line, leading blanks are not removed either. This lets you interactively enter '`I \showbox0␣`' to make TEX$_{GPC}$ show box 0. This doesn't work if the trailing blank were removed.

TEX writes an additional empty line whenever it prompts you on the terminal. TEX$_{GPC}$ doesn't. Finally TEX emits an '`Underfull \hbox`' warning whenever the last line of a paragraph happens to include glue only, because then TEX would erroneously remove the `parfillskip`. TEX$_{GPC}$ will keep it.

July 2014
Wolfgang Helbig, Programmierer `http://wwwlehre.dhbw-stuttgart.de/~helbig/`
Waiblingen, Baden-Württemberg `helbig.wolfgang@kabelbw.de`

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 protoTEX 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 `\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 TEX82 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 TEX82 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 TEX82 "frozen" from now on; stability and reliability are to be its main virtues.

On the other hand, the `WEB` description can be extended without changing the core of TEX82 itself, and the program has been designed so that such extensions are not extremely difficult to make. The *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].

h Since TEXGPC differs from TEX to make me happy, I have to change the banner line.

define *banner* ≡ ´This␣is␣TeX-GPC´

4.* The program begins with a normal Pascal program heading, whose components will be filled in later, using the conventions of WEB. 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 Pascal-H would ask the TEX user to specify a file name if *output* were specified here.

P Pascal wants the identifiers of the standard text files *input* and *output* in the parameterlist of the program header.

N One of the WEB macros is named *input* as well. To make TANGLE write INPUT into the Pascal source file instead of the expansion of the macro, you code the name as a concatenation of one letter identifiers since one letter identifiers cannot be macro names. The same applies to *type*.

G To access declarations from GPC's runtime system you need to **gpc_import** *gpc_gpc*. **gpc_only** avoids further name clashes.

N The procedure *initialize* passes *set_interrupt* to *gpc_install_signal_handler*. Since *set_interrupt* is declared further down, you need a *forward* declaration.

```
define term_in ≡ i@&n@&p@&u@&t
define term_out ≡ o@&u@&t@&p@&u@&t
define mtype ≡ t@&y@&p@&e
format mtype ≡ type
define gpc_import ≡ i@&m@&p@&o@&r@&t
format gpc_import ≡ label
define gpc_only ≡ o@&n@&l@&y
format gpc_only ≡ then
define gpc_gpc ≡ g@&p@&c
format mtype ≡ type  { 'mtype' will be equivalent to 'type' }
format type ≡ true  { but 'type' will not be treated as a reserved word }
⟨ Compiler directives 9* ⟩
program TEX(term_in, term_out);
  gpc_import gpc_gpc gpc_only (gpc_execute, gpc_install_signal_handler, gpc_sig_int);
  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 set_interrupt(signal : gpc_integer); forward;
  procedure initialize;  { this procedure gets things started properly }
    var ⟨ Local variables for initialization 19 ⟩
    begin ⟨ Initialize whatever TEX might access 8 ⟩
    end;
  ⟨ Basic printing procedures 57 ⟩
  ⟨ Error 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 '**debug** ... **gubed**', with apologies to people who wish to preserve the purity of English.

Similarly, there is some conditional code delimited by '**stat** ... **tats**' that is intended for use when statistics are to be kept about TEX's memory usage. The **stat** ... **tats** code also implements diagnostic information for `\tracingparagraphs` and `\tracingpages`.

define *debug* ≡ { change this to '*debug* ≡ `@{`' when not debugging }
define *gubed* ≡ { change this to '*gubed* ≡ `@}`' when not debugging }
format *debug* ≡ *begin*
format *gubed* ≡ *end*

define *stat* ≡ { change this to '*stat* ≡ `@{`' to turn off statistics }
define *tats* ≡ { change this to '*tats* ≡ `@}`' to turn off statistics }
format *stat* ≡ *begin*
format *tats* ≡ *end*

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 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.

G If the first character of a Pascal comment is a dollar sign, GNU Pascal treats the comment as a "compiler directive". GPC aborts when it detects an I/O error. To let TEX$_{\rm GPC}$ handle an I/O error while opening an input file, you have to turn off I/O checking altogether by the `I-` directive.

b In contrast to Pascal-H GNU Pascal offers no directive to check for arithmetic overflow.

e Knuth suggests to turn on range checking while debugging. GPC aborts when it spots range violation. Those violations might happen when the debugger shows a memeroy cell assumed to contain a *glue_ratio*. Even though turning of range checking doubles the speed of TEX I suggest to turn it on when not debugging, just to get another check from Knuth for discovering an error.

⟨ Compiler directives 9* ⟩ ≡
`@{@&$`*I*`−@}` { no I/O checking }
debug `@{@&$`*R*`−@}` **gubed** { no range check while debugging }

This code is used in section 4*.

10.* This TEX implementation conforms to the rules of the *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 **case** statements may include a default case that applies if no matching label is found. Thus, we shall use constructions like

case x **of**
1: ⟨code for $x = 1$⟩;
3: ⟨code for $x = 3$⟩;
othercases ⟨code for $x \neq 1$ and $x \neq 3$⟩
endcases

since most Pascal compilers have plugged this hole in the language by incorporating some sort of default mechanism. For example, the Pascal-H compiler allows '*others*:' as a default label, and other Pascals allow syntaxes like '**else**' or '**otherwise**' or '*otherwise*:', etc. The definitions of **othercases** and **endcases** should be changed to agree with local conventions. Note that no semicolon appears before **endcases** in this program, so the definition of **endcases** should include a semicolon if the compiler wants one. (Of course, if no default mechanism is available, the **case** statements of 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!)

G This is the only place I voluntarily use a GPC extension to Pascal. GPC offers *otherwise* and **else** . I decided for **else** because I do not want to add another gpc-keyword. Furthermore, Wirth uses **else** in Modula 2.

define *othercases* ≡ **else** {default for cases not listed explicitly}
define *endcases* ≡ **end** {follows the default case in an extended **case** statement}
format *othercases* ≡ *else*
format *endcases* ≡ *end*

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.

N I didn't change the constants, leaving it up to you to adopt them to your task. Just keep the Pascal source for `tex` around. So you can change them without going all the way through modifying `tex.ch` and tangeling it. Note that for `initex` *mem_top* and *mem_max* must agree.

U One of the constants is the filename of the string pool file which needs an adoption to Unix.

⟨ Constants in the outer block 11* ⟩ ≡

mem_max = 30000;
{ greatest index in TEX's internal *mem* array; must be strictly less than *max_halfword*; must be equal to *mem_top* in `INITEX`, otherwise ≥ *mem_top* }
mem_min = 0; { smallest index in TEX's internal *mem* array; must be *min_halfword* or more; must be equal to *mem_bot* in `INITEX`, otherwise ≤ *mem_bot* }
buf_size = 500; { maximum number of characters simultaneously present in current lines of open files and in control sequences between `\csname` and `\endcsname`; must not exceed *max_halfword* }
error_line = 72; { width of context lines on terminal error messages }
half_error_line = 42; { width of first lines of contexts in terminal error messages; should be between 30 and *error_line* − 15 }
max_print_line = 79; { width of longest text lines output; should be at least 60 }
stack_size = 200; { maximum number of simultaneous input sources }
max_in_open = 6;
{ maximum number of input files and error insertions that can be going on simultaneously }
font_max = 75; { maximum internal font number; must not exceed *max_quarterword* and must be at most *font_base* + 256 }
font_mem_size = 20000; { number of words of *font_info* for all fonts }
param_size = 60; { maximum number of simultaneous macro parameters }
nest_size = 40; { maximum number of semantic levels simultaneously active }
max_strings = 3000; { maximum number of strings; must not exceed *max_halfword* }
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 }
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 *string_vacancies* by the total length of TEX's own strings, which is currently about 23000 }
save_size = 600; { space for saving values outside of current group; must be at most *max_halfword* }
trie_size = 8000; { space for hyphenation patterns; should be larger for `INITEX` than it is in production versions of TEX }
trie_op_size = 500; { space for "opcodes" in the hyphenation patterns }
dvi_buf_size = 800; { size of the output buffer; must be a multiple of 8 }
file_name_size = 40; { file names shouldn't be longer than this }
pool_name = ´`TeXformats/tex.pool␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣`´;
{ string of length *file_name_size*; tells where the string pool appears }

This code is used in section 4*.

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 *alpha_file* for a file that contains textual data, and the term *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 *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.

G GNU Pascal ignores **packed** for file types. Integer subranges occupy 32 bits, so it writes 4 byte for every *eight_bits* element. GNU Pascal offers two extensions to get at 8 bit bytes: Use the predefined type *byte* for *eight_bits* or pack the subrange type. Since packing subrange types is rather strange extension to Pascal, I decided for the byte.

define *gpc_byte* ≡ *b@&y@&t@&e*

⟨ Types in the outer block 18 ⟩ +≡

eight_bits = *gpc_byte*; { unsigned one-byte quantity }

P *alpha_file* = *t@&e@&x@&t*; { Pascal requires *text* }

byte_file = **packed file of** *eight_bits*; { files that contain binary data }

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 f, we can write

reset$(f, name, \text{'/0'})$	for input;
rewrite$(f, name, \text{'/0'})$	for output.

The '*name*' parameter, which is of type '**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 *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 $erstat(f) \neq 0$ after an unsuccessful *reset* or *rewrite*. This allows TEX to undertake appropriate corrective action.

G In Pascal, external files must occur in the program heading and GNU Pascal asks the user whenever an external file is opened. But `initex` wants to reset `tex.pool` and rewrite `plain.fmt` without asking the user for the file name. We are lucky: GNU Pascal lets you open external files by passing its name as a second argument to *reset* resp. *rewrite*. The function *gpc_trim* removes trailing spaces that would otherwise be part of the file name. The function *gpc_io_result* returns a nonzero value if any error occurred since the last invocation of *gpc_io_result*.

G Buffering output of the DVI file accelerates TEXGPC dramatically. To get at output buffering, the file needs to be a *gpc_untyped_file*;

```
define gpc_trim ≡ t@&r@&i@&m
define gpc_io_result ≡ i@&o@&r@&e@&s@&u@&l@&t
define reset_OK(#) ≡ gpc_io_result = 0
define rewrite_OK(#) ≡ gpc_io_result = 0
define clear_io_result ≡ if gpc_io_result = 0 then do_nothing
function a_open_in(var f : alpha_file): boolean; { open a text file for input }
  begin clear_io_result; reset(f, gpc_trim(name_of_file)); a_open_in ← reset_OK(f);
  end;
function a_open_out(var f : alpha_file): boolean; { open a text file for output }
  begin clear_io_result; rewrite(f, gpc_trim(name_of_file)); a_open_out ← rewrite_OK(f);
  end;
function b_open_in(var f : byte_file): boolean; { open a binary file for input }
  begin clear_io_result; reset(f, gpc_trim(name_of_file)); b_open_in ← reset_OK(f);
  end;
function b_open_out(var f : gpc_untyped_file): boolean; { open a binary file for output }
  begin clear_io_result; rewrite(f, gpc_trim(name_of_file), 1); b_open_out ← rewrite_OK(f);
  end;
function w_open_in(var f : word_file): boolean; { open a word file for input }
  begin clear_io_result; reset(f, gpc_trim(name_of_file)); w_open_in ← reset_OK(f);
  end;
function w_open_out(var f : word_file): boolean; { open a word file for output }
  begin clear_io_result; rewrite(f, gpc_trim(name_of_file)); w_open_out ← rewrite_OK(f);
  end;
```

28.* Files can be closed with the Pascal-H routine '*close*(*f*)', which should be used when all input or output with respect to *f* has been completed. This makes *f* available to be opened again, if desired; and if *f* was used for output, the *close* operation makes the corresponding external file appear on the user's area, ready to be read.

These procedures should not generate error messages if a file is being closed before it has been successfully opened.

G GNU Pascal has accidently a very similar procedure *gpc_close*. But we need another routine to close a *gpc_untyped_file*, which is necessary for buffering the output.

```
define gpc_close ≡ c@&l@&o@&s@&e
procedure a_close(var f : alpha_file); { close a text file }
  begin close(f);
  end;
procedure b_close(var f : byte_file); { close a binary file }
  begin close(f);
  end;
procedure w_close(var f : word_file); { close a word file }
  begin close(f);
  end;
procedure u_close(var f : gpc_untyped_file); { close an untyped file }
  begin close(f);
  end;
```

31.* The *input_ln* function brings the next line of input from the specified file into available positions of the buffer array and returns the value *true*, unless the file has already been entirely read, in which case it returns *false* and sets $last \leftarrow first$. In general, the *ASCII_code* numbers that represent the next line of the file are input into $buffer[first]$, $buffer[first+1]$, ..., $buffer[last-1]$; 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 $buffer[last-1] \neq$ "␣".

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

The variable *max_buf_stack*, which is used to keep track of how large the *buf_size* parameter must be to accommodate the present job, is also kept up to date by *input_ln*.

If the *bypass_eoln* parameter is *true*, *input_ln* will do a *get* before looking at the first character of the line; this skips over an *eoln* that was in $f\uparrow$. The procedure does not do a *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 *eoln* immediately before *eof*, but TEX needs only a weaker restriction: If *eof* occurs in the middle of a line, the system function *eoln* should return a *true* result (even though $f\uparrow$ will be undefined).

Since the inner loop of *input_ln* is part of TEX's "inner loop"—each character of input comes in at this place—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.

h Since TEXGPC does not remove trailing spaces, $buffer[last-1]$ might hold a space.

P Pascal-H lets you *reset* the terminal input file with the first *get* 'surpressed'. For several reasons, this feature is not exploited by TEXGPC. First, it is not provided by GPC. Second rightly so, since it violates the specification of Pascal. Third, it makes the program quite ugly by destroying the beautiful equivalence of terminal and disk files. Fourth, since TEXGPC uses Pascal's standard text file *input*, it should not reset that file at all. Fifth, surpressing the first *get* is offered by Pascal-H to address a problem, namely that the program stays in the reset function waiting for user input and this problem is solved much more beautiful by "lazy I/O", whereby the program only waits for user input if it is needed. This is suggested in the *Pascal User Manual*, implemented by GNU Pascal and exploited by TEXGPC. This leads to a much cleaner implementaion of *input_ln*, which can always savely assume that $f\uparrow$ holds the first character of the next line. This condition is established by Pascal's *reset* and maintained by *input_ln*.

h Unlike TEX82 TEXGPC leaves trailing spaces in the input line.

G Frank Heckenbach pointed out that GNU Pascal employes buffered I/O on input files—no need to avoid high system overhead here.

```
function input_ln(var f : alpha_file; bypass_eoln : boolean): boolean;
        { inputs the next line or returns false }
  begin   { ignore bypass_eoln. Assuming f being positioned at the first character }
  last ← first;   { cf. Matthew 19:30 }
  if eof(f) then input_ln ← false
  else begin while ¬eoln(f) do
      begin if last ≥ max_buf_stack then
        begin max_buf_stack ← last + 1;
        if max_buf_stack = buf_size then
          begin read_ln(f);   { complete the current line }
          ⟨ Report overflow of the input buffer, and abort 35 ⟩;
          end;
        end;
      buffer[last] ← xord[f↑]; get(f); incr(last);
      end;
    get(f);   { Advance f to the first character of the next line }
```

```
    input_ln ← true;
    end;
  end;
```

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 *term_in*, and when it is considered an output file the file variable is *term_out*. Pascal's standard text files are declared implicitly.

P

33* Here is how to open the terminal files in Pascal-H. The '`/I`' switch suppresses the first *get*. In Pascal, the standard text files are openend implicitly.

P

```
define t_open_in ≡ do_nothing  { open the terminal for text input }
define t_open_out ≡ do_nothing  { open the terminal for text output }
```

34* 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.

G Nothing needs to be done to update the terminal, since GNU Pascal does not employ buffered output on typed files. I do not know how to clear the type ahead buffer, so TEXGPC does nothing here. Unix holds terminal output, when it receives `^S` and continues writing to the terminal, when it receives `^Q`. These 'flow **B** control' characters only work when sent from the terminal but not when sent to the terminal. Here I give up, since I don't know how to restart the output from the writing side so TEXGPC does nothing. Mac OS X does not stop terminal output when it receives `^S`.

```
define update_terminal ≡ do_nothing  { 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 '**'.)

F This procedure puts the command line arguments separated by spaces into *buffer*. Like *input_ln* it updates *last* so that *buffer*[*first* .. *last*) will contain the command line.

G GNU Pascal's function *gpc_param_count* gives the number of command line arguments. The function *gpc_param_str*(*n*) returns the n-th argument for $0 \le n \le$ *gpc_param_count* in a *gpc_string*, whose length is returned by the function *gpc_length*. A *gpc_string* is like a **packed array** [1 .. *gpc_length*] **of** *char* with varying length.

```
define loc ≡ cur_input.loc_field  { location of first unread character in buffer }
define gpc_string ≡ s@&t@&r@&i@&n@&g  { a string with varying length }
define gpc_length ≡ l@&e@&n@&g@&t@&h
define gpc_param_count ≡ p@&a@&r@&a@&m@&c@&o@&u@&n@&t
define gpc_param_str ≡ p@&a@&r@&a@&m@&s@&t@&r
        { GPC function returning the length of a gpc_string }
procedure input_command_ln;  { get the command line in buffer }
  var argc: integer;  { argument counter }
    arg: gpc_string;  { argument }
    cc: integer;  { character counter in argument }
  begin last ← first; argc ← 1;
  while argc ≤ gpc_param_count do
    begin cc ← 1; arg ← gpc_param_str(argc); incr(argc);
    while cc ≤ gpc_length(arg) do
      begin if last + 1 ≥ buf_size then ⟨Report overflow of the input buffer, and abort 35⟩;
      buffer[last] ← xord[arg[cc]]; incr(last); incr(cc);
      end;
    if (argc ≤ gpc_param_count) then
      begin buffer[last] ← "␣"; incr(last);  { insert a space between arguments }
      end;
    end;
  end;
```

F 37* The following program treats a non empty command line as the first line.

The 2008 edition of TEXGPC erranously assumed $buffer[last-1] \neq$ "␣" which does not hold if your first line is all blank as Joachim Kuebart noted.

```
function init_terminal: boolean; { gets the terminal input started }
  label exit;
  begin t_open_in; input_command_ln;
  while first = last do
    begin wake_up_terminal; write(term_out, '**'); update_terminal;
    if ¬input_ln(term_in, true) then { this shouldn't happen }
      begin write_ln(term_out); write_ln(term_out, '! End of file on the terminal... why?');
      init_terminal ← false; return;
      end;
    if first = last then write_ln(term_out, 'Please type the name of your input file.');
    end;
  loc ← first; init_terminal ← true;
exit: end;
```

79:* Individual lines of help are recorded in the array *help_line*, which contains entries in positions 0 .. (*help_ptr* − 1). They should be printed in reverse order, i.e., with *help_line*[0] appearing last.

F TEXGPC lets the user jump into `vi` to edit the current input file at the current line. After saveing line number and file name TEXGPC jumps out and then launches `vi` passing the saved values.

```
define hlp1(#) ≡ help_line[0] ← #; end
define hlp2(#) ≡ help_line[1] ← #; hlp1
define hlp3(#) ≡ help_line[2] ← #; hlp2
define hlp4(#) ≡ help_line[3] ← #; hlp3
define hlp5(#) ≡ help_line[4] ← #; hlp4
define hlp6(#) ≡ help_line[5] ← #; hlp5
define help0 ≡ help_ptr ← 0  { sometimes there might be no help }
define help1 ≡ begin help_ptr ← 1; hlp1  { use this with one help line }
define help2 ≡ begin help_ptr ← 2; hlp2  { use this with two help lines }
define help3 ≡ begin help_ptr ← 3; hlp3  { use this with three help lines }
define help4 ≡ begin help_ptr ← 4; hlp4  { use this with four help lines }
define help5 ≡ begin help_ptr ← 5; hlp5  { use this with five help lines }
define help6 ≡ begin help_ptr ← 6; hlp6  { use this with six help lines }
```

⟨ Global variables 13 ⟩ +≡

```
help_line: array [0 .. 5] of str_number;  { helps for the next error }
help_ptr: 0 .. 6;  { the number of help lines present }
use_err_help: boolean;  { should the err_help list be shown? }
edit_line: integer;  { line number to be passed to the system editor }
edit_file_name: str_number;  { file name to be passed to the system editor }
```

80:* ⟨ Set initial values of key variables 21 ⟩ +≡

```
help_ptr ← 0; use_err_help ← false; edit_line ← 0; edit_file_name ← 0;
```

84:* 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.

⟨ Interpret code *c* and **return** if done 84* ⟩ ≡

```
case c of
"0","1","2","3","4","5","6","7","8","9": if deletions_allowed then
    ⟨ Delete c − "0" tokens and goto continue 88 ⟩;
debug "D": begin debug_help; goto continue; end; gubed
"E": if base_ptr > 0 then
F     begin  { save values to be passed to the system editor }
      edit_file_name ← input_stack[base_ptr].name_field; edit_line ← line; interaction ← scroll_mode;
      jump_out;
      end;
"H": ⟨ Print the help information and goto continue 89 ⟩;
"I": ⟨ Introduce new material from the terminal and return 87 ⟩;
"Q","R","S": ⟨ Change the interaction level and return 86 ⟩;
"X": begin interaction ← scroll_mode; jump_out;
  end;
othercases do_nothing
endcases;
⟨ Print the menu of available options 85 ⟩
```

This code is used in section 83.

96.* Users occasionally want to interrupt TEX while it's running. If the Pascal runtime system allows this, one can implement a routine that sets the global variable *interrupt* to some nonzero value when such an interrupt is signalled. Otherwise there is probably at least a way to make *interrupt* nonzero using the Pascal debugger.

G GNU Pascal reserves the identifier *interrupt*, which seems a bug. `WEB` provides a simple workaround.

```
define interrupt ≡ tex_interrupt
format interrupt ≡ true
define check_interrupt ≡
        begin if interrupt ≠ 0 then pause_for_instructions;
        end
```

⟨ Global variables 13 ⟩ +≡
interrupt: *integer*; { should TEX pause for instructions? }
OK_to_interrupt: *boolean*; { should interrupts be observed? }

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.)

G In GNU Pascal a *gpc_short_real* has the desired size.

```
define gpc_short_real ≡ s@&h@&o@&r@&t@&r@&e@&a@&l
define set_glue_ratio_zero(#) ≡ # ← 0.0   { store the representation of zero ratio }
define set_glue_ratio_one(#) ≡ # ← 1.0   { store the representation of unit ratio }
define float(#) ≡ #   { convert from glue_ratio to type real }
define unfloat(#) ≡ #   { convert from real to type glue_ratio }
define float_constant(#) ≡ #.0   { convert integer constant to real }
```

⟨ Types in the outer block 18 ⟩ +≡

```
glue_ratio = gpc_short_real;   { one-word representation of a glue expansion factor in GNU 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* $= 0$. And this can be done here! G

define *qi*(#) ≡ # {to put an *eight_bits* item into a quarterword}
define *qo*(#) ≡ # {to take an *eight_bits* item out of a quarterword}
define *hi*(#) ≡ # {to put a sixteen-bit item into a halfword}
define *ho*(#) ≡ # {to take a sixteen-bit item from a halfword}

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 specifies July 4, 1776, at noon; but users probably want a better approximation to the truth.

G GNU Pascal provides the *gpc_get_time_stamp* function, which stores the system time in its argument.

```
define gpc_time_stamp ≡ t@&i@&m@&e@&s@&t@&a@&m@&p
define gpc_get_time_stamp ≡ g@&e@&t@&t@&i@&m@&e@&s@&t@&a@&m@&p
define gpc_minute ≡ m@&i@&n@&u@&t@&e
define gpc_hour ≡ h@&o@&u@&r
define gpc_day ≡ d@&a@&y
define gpc_month ≡ m@&o@&n@&t@&h
define gpc_year ≡ y@&e@&a@&r
procedure fix_date_and_time;
  var t: gpc_time_stamp;
  begin gpc_get_time_stamp(t); time ← t.gpc_minute + t.gpc_hour * 60; { minutes since midnight }
  day ← t.gpc_day; month ← t.gpc_month; year ← t.gpc_year; { Anno Domini }
  end;
```

360*. All of the easy branches of *get_next* have now been taken care of. There is one more branch.

h TEX82 ends the current line by calling *print_ln* even if the line is empty. This causes an additional empty line that I want to avoid. Calling *print_nl*("") is smarter. It ends the current line only if it is not empty.

```
define end_line_char_inactive ≡ (end_line_char < 0) ∨ (end_line_char > 255)

⟨ Move to next line of file, or goto restart if there is no next line, or return if a \read line has
    finished 360* ⟩ ≡
  if name > 17 then ⟨ Read next line of file into buffer, or goto restart if the file has ended 362 ⟩
  else begin if ¬terminal_input then { \read line has ended }
      begin cur_cmd ← 0; cur_chr ← 0; return;
      end;
    if input_ptr > 0 then { text was inserted during error recovery }
      begin end_file_reading; goto restart; { resume previous level }
      end;
    if selector < log_only then open_log_file;
    if interaction > nonstop_mode then
      begin if end_line_char_inactive then incr(limit);
      if limit = start then { previous line was empty }
        print_nl("(Please␣type␣a␣command␣or␣say␣`\end´)");
      print_nl(""); first ← start; prompt_input("*"); { input on-line into buffer }
      limit ← last;
      if end_line_char_inactive then decr(limit)
      else buffer[limit] ← end_line_char;
      first ← limit + 1; loc ← start;
      end
    else fatal_error("***␣(job␣aborted,␣no␣legal␣\end␣found)");
        { nonstop mode, which is intended for overnight batch processing, never waits for on-line input }
    end
```

This code is used in section 343.

514.* Input files that can't be found in the user's area may appear in a standard system area called *TEX_area*. Font metric files whose areas are not given explicitly are assumed to appear in a standard system area called *TEX_font_area*. These system area names will, of course, vary from place to place.

U Use the Unix file separator.

```
define TEX_area ≡ "TeXinputs/"  { i.e., a subdirectory of the working directory }
define TEX_font_area ≡ "TeXfonts/"  { dito }
```

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.

```
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 }
U   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;
```

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

U *TEX_format_default* ← ´TeXformats/plain.fmt´; { "/" is the Unix file name separator }

532.* Here's an example of how these conventions are used. Whenever it is time to ship out a box of stuff, we shall use the macro *ensure_dvi_open*.

G To get buffered output, the file needs to be a *gpc_untyped_file*.

```
define gpc_untyped_file ≡ f@&i@&l@&e
define ensure_dvi_open ≡
        if output_file_name = 0 then
          begin if job_name = 0 then open_log_file;
          pack_job_name(".dvi");
          while ¬b_open_out(dvi_file) do prompt_file_name("file name for output", ".dvi");
          output_file_name ← make_name_string;
          end
```

⟨ Global variables 13 ⟩ +≡

```
dvi_file: gpc_untyped_file;  { the device-independent output goes here }
output_file_name: str_number;  { full name of the output file }
log_name: str_number;  { full name of the log file }
```

537* Let's turn now to the procedure that is used to initiate file reading when an '`\input`' command is being processed.

e Keep the complete file name since it might be needed to be passed to the system editor. (TEX82 strips off area and extension to conserve string pool space.)

```
procedure start_input; { TEX will \input something }
  label done;
  begin scan_file_name; { set cur_name to desired file name }
  if cur_ext = "" then cur_ext ← ".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 ← a_make_name_string(cur_file);
  if job_name = 0 then
    begin job_name ← 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_ln
  else if (term_offset > 0) ∨ (file_offset > 0) then print_char("␣");
  print_char("("); incr(open_parens); slow_print(name); update_terminal; state ← new_line;
  ⟨Read the first line of the new file 538⟩;
  end;
```

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.

G In fact, buffering dramatically cuts down system overhead. To compile this document, a program without buffering spent 48.45 s in the kernel but with buffering only 0.57 s. The total times were 64.45 s vs. 11.40 s.

GNU Pascal's *gpc_block_write* procedure takes an untyped **file** , an **array** and the number of bytes to be written. The **array** here is given as a 'slice', another extension of GNU Pascal. It should be clear, what *buffer*[*a* .. *b*] means. This simple change was suggested by Emil Jerabek.

```
define gpc_block_write ≡ b@&l@&o@&c@&k@&w@&r@&i@&t@&e
procedure write_dvi(a, b : dvi_index);
  begin gpc_block_write(dvi_file, dvi_buf[a .. b], b − a + 1);
  end;
```

642* At the end of the program, we must finish things off by writing the postamble. If *total_pages* = 0, the DVI file was never opened. If *total_pages* ≥ 65536, the DVI file will lie. And if *max_push* ≥ 65536, the user deserves whatever chaos might ensue.

An integer variable *k* will be declared for use by this routine.

```
⟨ Finish the DVI file 642* ⟩ ≡
  while cur_s > −1 do
    begin if cur_s > 0 then dvi_out(pop)
    else begin dvi_out(eop); incr(total_pages);
      end;
    decr(cur_s);
    end;
  if total_pages = 0 then print_nl("No␣pages␣of␣output.")
  else begin dvi_out(post); { beginning of the postamble }
    dvi_four(last_bop); last_bop ← dvi_offset + dvi_ptr − 5; { post location }
    dvi_four(25400000); dvi_four(473628672); { conversion ratio for sp }
    prepare_mag; dvi_four(mag); { magnification factor }
    dvi_four(max_v); dvi_four(max_h);
    dvi_out(max_push div 256); dvi_out(max_push mod 256);
    dvi_out((total_pages div 256) mod 256); dvi_out(total_pages mod 256);
    ⟨ Output the font definitions for all fonts that were used 643 ⟩;
    dvi_out(post_post); dvi_four(last_bop); dvi_out(id_byte);
    k ← 4 + ((dvi_buf_size − dvi_ptr) mod 4); { the number of 223's }
    while k > 0 do
      begin dvi_out(223); decr(k);
      end;
    ⟨ Empty the last bytes out of dvi_buf 599 ⟩;
    print_nl("Output␣written␣on␣"); slow_print(output_file_name); print("␣("); print_int(total_pages);
    print("␣page");
    if total_pages ≠ 1 then print_char("s");
    print(",␣"); print_int(dvi_offset + dvi_ptr); print("␣bytes)."); u_close(dvi_file);
G       { dvi_file is an untyped file }
    end
```

This code is used in section 1333*.

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 '`$$`'. The *par_fill_skip* is preceded by an infinite penalty, so it will never be considered as a potential breakpoint.

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

E TEX82 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, TEXGPC saves a pointer to the node immediately preceding the `\parfillskip` node and quits pruning when it encounters this node several procedures later.

```
⟨ Get ready to start line breaking 816* ⟩ ≡
  link(temp_head) ← link(head);
  if is_char_node(tail) then tail_append(new_penalty(inf_penalty))
  else if type(tail) ≠ glue_node then tail_append(new_penalty(inf_penalty))
    else begin type(tail) ← penalty_node; delete_glue_ref(glue_ptr(tail)); flush_node_list(leader_ptr(tail));
      penalty(tail) ← inf_penalty;
      end;
  non_prunable_p ← tail; { points to the node immediately before \parfillskip }
  link(tail) ← new_param_glue(par_fill_skip_code); init_cur_lang ← prev_graf mod ′200000;
  init_l_hyf ← prev_graf div ′20000000; init_r_hyf ← (prev_graf div ′200000) mod ′100; pop_nest;
```

See also sections 827, 834, and 848.

This code is used in section 815.

862* Breaking paragraphs into lines, continued. So far we have gotten a little way into the *line_break* routine, having covered its important *try_break* subroutine. Now let's consider the rest of the process.

The main loop of *line_break* traverses the given hlist, starting at *link*(*temp_head*), and calls *try_break* at each legal breakpoint. A variable called *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 *cur_p*. Another variable, *prev_p*, is usually one step behind *cur_p*, but the real meaning of *prev_p* is this: If *type*(*cur_p*) = *glue_node* then *cur_p* is a legal breakpoint if and only if *auto_breaking* is true and *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.

E Declare the *non_prunable_p* pointer.

⟨ Local variables for line breaking 862* ⟩ ≡
auto_breaking: *boolean*; { is node *cur_p* outside a formula? }
non_prunable_p: *pointer*; { pointer to the node before `\parfillskip` }
prev_p: *pointer*; { helps to determine when glue nodes are breakpoints }
q, *r*, *s*, *prev_s*: *pointer*; { miscellaneous nodes of temporary interest }
f: *internal_font_number*; { used when calculating character widths }

See also section 893.

This code is used in section 815.

876* Once the best sequence of breakpoints has been found (hurray), we call on the procedure *post_line_break* to finish the remainder of the work. (By introducing this subprocedure, we are able to keep *line_break* from getting extremely long.)

E Pass *non_prunable_p* to the *post_line_break* procedure.

⟨ Break the paragraph at the chosen breakpoints, justify the resulting lines to the correct widths, and append them to the current vertical list 876* ⟩ ≡
post_line_break(*final_widow_penalty*, *non_prunable_p*)

This code is used in section 815.

877* The total number of lines that will be set by *post_line_break* is *best_line* − *prev_graf* − 1. The last breakpoint is specified by *break_node*(*best_bet*), and this passive node points to the other breakpoints via the *prev_break* links. The finishing-up phase starts by linking the relevant passive nodes in forward order, changing *prev_break* to *next_break*. (The *next_break* fields actually reside in the same memory space as the *prev_break* fields did, but we give them a new name because of their new significance.) Then the lines are justified, one by one.

E Declare another parameter. It holds the pointer to the node immediately preceding `\parfillskip`.

```
define next_break ≡ prev_break  { new name for prev_break after links are reversed }
⟨ Declare subprocedures for line_break 826 ⟩ +≡
procedure post_line_break(final_widow_penalty : integer; non_prunable_p : pointer);
  label done, done1;
  var q, r, s: pointer;  { temporary registers for list manipulation }
    disc_break: boolean;  { was the current break at a discretionary node? }
    post_disc_break: boolean;  { and did it have a nonempty post-break part? }
    cur_width: scaled;  { width of line number cur_line }
    cur_indent: scaled;  { left margin of line number cur_line }
    t: quarterword;  { used for replacement counts in discretionary nodes }
    pen: integer;  { use when calculating penalties between lines }
    cur_line: halfword;  { the current line number being justified }
  begin ⟨ Reverse the links of the relevant passive nodes, setting cur_p to the first breakpoint 878 ⟩;
  cur_line ← prev_graf + 1;
  repeat ⟨ Justify the line ending at breakpoint cur_p, and append it to the current vertical list, together
      with associated penalties and other insertions 880 ⟩;
    incr(cur_line); cur_p ← next_break(cur_p);
    if cur_p ≠ null then
      if ¬post_disc_break then ⟨ Prune unwanted nodes at the beginning of the next line 879* ⟩;
  until cur_p = null;
  if (cur_line ≠ best_line) ∨ (link(temp_head) ≠ null) then confusion("line␣breaking");
  prev_graf ← best_line − 1;
  end;
```

879* 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.

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

```
⟨ Prune unwanted nodes at the beginning of the next line 879* ⟩ ≡
  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 877*.

1332.* Now this is really it: TEX starts and ends here.

The initial test involving *ready_already* should be deleted if the Pascal runtime system is smart enough to detect such a "mistake."

h TEXGPC tries to load the format file even before it initializes the output routines. That way, it will print the name of the format file on the terminal.

G *gpc_execute* starts the system editor (vi) and *gpc_halt* passes the *history* as an exit code to the shell.

```
define gpc_halt ≡ h@&a@&l@&t
begin { start_here }
history ← 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 "constant" values for consistency 14 ⟩
if bad > 0 then
  begin wterm_ln('Ouch---my␣internal␣constants␣have␣been␣clobbered!', '---case␣', bad : 1);
  goto final_end;
  end;
initialize; { set global variables to their starting values }
init if ¬get_strings_started then goto final_end;
init_prim; { call primitive for each primitive }
init_str_ptr ← str_ptr; init_pool_ptr ← pool_ptr; fix_date_and_time;
tini
ready_already ← 314159;
start_of_TEX: ⟨ Preload the default format file 1379* ⟩;
⟨ Initialize the output routines 55 ⟩;
⟨ Get the first line of input and prepare to start 1337 ⟩;
history ← spotless; { ready to go! }
main_control; { come to life }
final_cleanup; { prepare for death }
end_of_TEX: close_files_and_terminate;
```

F `final_end: if edit_file_name > 0 then start_editor; { user typed 'E' }`

u `gpc_halt(history); { pass history as the exit value to the system }`

`end.`

1333.* 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 *str_room* or *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 *prepare_mag*; but that can't cause infinite recursion.

This program doesn't bother to close the input files that may still be open.

h Special care is taken to terminate the last line on the terminal.

```
⟨ Last-minute procedures 1333* ⟩ ≡
procedure close_files_and_terminate;
  var k: integer; { all-purpose index }
  begin ⟨ Finish the extensions 1378 ⟩;
  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(".");
      end;
    end;
  if term_offset > 0 then wterm_cr;
  end;
```

See also sections 1335, 1336, and 1338*.

This code is used in section 1330.

1338.* Debugging. 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.)

P 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.

```
define breakpoint = 888  { place where a breakpoint is desirable }
⟨ Last-minute procedures 1333* ⟩ +≡
  debug procedure debug_help;  { routine to display various things }
  label breakpoint, exit;
  var k, l, m, n: integer;
  begin loop
    begin ; wake_up_terminal; print_nl("debug␣#␣(-1␣to␣exit):"); update_terminal;
    if eof(term_in) then return;
    read(term_in, m);
    if m < 0 then return
    else if m = 0 then
        begin goto breakpoint; @\  { go to every label at least once }
      breakpoint: m ← 0; @{´BREAKPOINT´@}@\
        end
      else begin if eof(term_in) then return;
        read(term_in, n);
        case m of
        ⟨ Numbered cases for debug_help 1339* ⟩
        othercases print("?")
        endcases;
        end;
    end;
exit: end;
  gubed
```

1339.* ⟨ Numbered cases for *debug_help* 1339* ⟩ ≡

```
1: print_word(mem[n]);  { display mem[n] in all forms }
2: print_int(info(n));
3: print_int(link(n));
4: print_word(eqtb[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 ← 10000; depth_threshold ← 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 }
13: begin if eof(term_in) then return;
   read(term_in, l); print_cmd_chr(n, l);
   end;
14: for k ← 0 to n do print(buffer[k]);
15: begin font_in_short_display ← null_font; short_display(n);
   end;
16: panicking ← ¬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.

h Try to preload the default format file. This is called even before the first line is read from the terminal, and thus turns `VIRTEX` into `TEX`, at least as experienced by the user. `INITEX` sets *format_ident* to '`INITEX`' and won't load a format file here.

```
⟨ Preload the default format file 1379* ⟩ ≡
  if format_ident = 0 then
    begin pack_buffered_name(format_default_length − format_ext_length, 1, 0);
    if ¬w_open_in(fmt_file) then
      begin wterm_ln('I␣can''t␣find␣the␣format␣file␣', name_of_file); goto final_end
      end;
    if ¬load_fmt_file then
      begin w_close(fmt_file); goto final_end
      end;
    w_close(fmt_file);
    end
```

This code is used in section 1332*.

F **1380*** If the user typed `'E'` to edit a file after confronted with an error message, TEX will clean up and then call *start_editor* as its last feat. The file name and line number to be passed to the system editor are saved in *edit_file_name* and *edit_line*.

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

Beware of using any `WEB` strings like `"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 *name_of_file* cannot happen, since *name_of_file* kept the file name while the file was being **G** opened. The *gpc_write_str* function writes its arguments into a *gpc_string* to build the command line. The function *gpc_execute* takes a *gpc_string* which holds the command line to be executed.

```
define gpc_execute ≡ e@&x@&e@&c@&u@&t@&e
define gpc_write_str ≡ w@&r@&i@&t@&e@&s@&t@&r
```

```
⟨ Error handling procedures 78 ⟩ +≡
procedure start_editor;
  var i: integer; { index into name_of_file }
    j: pool_pointer; { index into str_pool }
    cmd_line: gpc_string(200); { area to build the command line }
  begin i ← 1; j ← str_start[edit_file_name];
  while j < str_start[edit_file_name + 1] do
    begin name_of_file[i] ← xchr[str_pool[j]]; incr(i); incr(j)
    end;
  while i ≤ file_name_size do
    begin name_of_file[i] ← '␣'; incr(i)
    end;
  gpc_write_str(cmd_line, 'vi␣+', line, '␣', gpc_trim(name_of_file));
  if 0 ≠ gpc_execute(cmd_line) then
    write_ln(gpc_param_str(0), ':␣could␣not␣start␣editor␣with:␣"', cmd_line, '"');
  end;
```

F **1381.*** The next modules declare and install the interrupt procedure *set_interrupt*.

The identifiers are truncated by `TANGLE` to twelve characters. We use this trick to persuade `TANGLE` to transfer the complete name to the Pascal source.

define *gpc_install_signal_handler* ≡ *i@&n@&s@&t@&a@&l@&l@&s@&i@&g@&n@&a@&l@&h@&a@&n@&d@&l@&e@&r*
define *gpc_sig_int* ≡ *s@&i@&g@&i@&n@&t*
define *gpc_null* ≡ *n@&u@&l@&l*
define *gpc_integer* ≡ *cinteger* { for earlier versions of GPC (3.2) replace *cinteger* by *integer* }

⟨ Error handling procedures 78 ⟩ +≡
procedure *set_interrupt*(*signal* : *gpc_integer*);
begin *interrupt* ← 1 **end**;

1382.* To install *set_interrupt* as our 'signal handler', I use the procedure *gpc_install_signal_handler*. It works with these arguments, but don't ask why. GNU Pascal's *gpc_sig_int* constant denotes the Unix interrupt signal, which is sent when the user types `^C`. Then *set_interrupt* is called, which sets the global variable *interrupt* to one, thus causing TEX to invoke *error* to engage the user in an error dialog.

The 2008 edition had a bug: It treated the function *gpc_install_signal_handler* as a procedure. The old GPC version didn't care, but the current one does. This was discovered by Luis Rivera and Martin Monperrus.

⟨ Initialize whatever TEX might access 8 ⟩ +≡
if *gpc_install_signal_handler*(*gpc_sig_int*, *set_interrupt*, *true*, *true*, *gpc_null*, *gpc_null*) **then** *do_nothing*;

1383.* 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, 25, 27, 28, 31, 32, 33, 34, 36, 37, 79, 80, 84, 96, 109, 112, 241, 360, 514, 516, 521, 532, 537, 597, 642, 816, 862, 876, 877, 879, 1332, 1333, 1338, 1339, 1379, 1380, 1381, 1382, 1383.

`**`: 37,* 534.
`*`: 174, 176, 178, 313, 360,* 856, 1006, 1355.
`->`: 294.
`=>`: 363.
`???`: 59.
`?`: 83.
`@`: 856.
`@@`: 846.
a: 47, 102, 218, 518, 519, 523, 560, 597,* 691, 722, 738, 752, 1123, 1194, 1211, 1236, 1257.
`A <box> was supposed to...`: 1084.
a_close: 28,* 51, 329, 485, 486, 1275, 1333,* 1374, 1378.
a_leaders: 149, 189, 625, 627, 634, 636, 656, 671, 1071, 1072, 1073, 1078, 1148.
a_make_name_string: 525, 534, 537.*
a_open_in: 27,* 51, 537,* 1275.
a_open_out: 27,* 534, 1374.
A_token: 445.
abort: 560, 563, 564, 565, 568, 569, 570, 571, 573, 575.
above: 208, 1046, 1178, 1179, 1180.
`\above` primitive: 1178.
above_code: 1178, 1179, 1182, 1183.
above_display_short_skip: 224, 814.
`\abovedisplayshortskip` primitive: 226.
above_display_short_skip_code: 224, 225, 226, 1203.
above_display_skip: 224, 814.
`\abovedisplayskip` primitive: 226.
above_display_skip_code: 224, 225, 226, 1203, 1206.
`\abovewithdelims` primitive: 1178.
abs: 66, 186, 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.
absorbing: 305, 306, 339, 473.
acc_kern: 155, 191, 1125.
accent: 208, 265, 266, 1090, 1122, 1164, 1165.
`\accent` primitive: 265.
accent_chr: 687, 696, 738, 1165.
accent_noad: 687, 690, 696, 698, 733, 761, 1165, 1186.
accent_noad_size: 687, 698, 761, 1165.
act_width: 866, 867, 868, 869, 871.
action procedure: 1029.
active: 162, 819, 829, 843, 854, 860, 861, 863, 864, 865, 873, 874, 875.
active_base: 220, 222, 252, 253, 255, 262, 263, 353, 442, 506, 1152, 1257, 1289, 1315, 1317.
active_char: 207, 344, 506.
active_height: 970, 975, 976.
active_node_size: 819, 845, 860, 864, 865.
active_width: 823, 824, 829, 843, 861, 864, 866, 868, 970.
actual_looseness: 872, 873, 875.
add_delims_to: 347.
add_glue_ref: 203, 206, 430, 802, 881, 996, 1100, 1229.
add_token_ref: 203, 206, 323, 979, 1012, 1016, 1221, 1227, 1357.
additional: 644, 645, 657, 672.
adj_demerits: 236, 836, 859.
`\adjdemerits` primitive: 238.
adj_demerits_code: 236, 237, 238.
adjust: 576.
adjust_head: 162, 888, 889, 1076, 1085, 1199, 1205.
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_tail: 647, 648, 649, 651, 655, 796, 888, 889, 1076, 1085, 1199.
adjusted_hbox_group: 269, 1062, 1083, 1085.
adv_past: 1362, 1363.
advance: 209, 265, 266, 1210, 1235, 1236, 1238.
`\advance` primitive: 265.
advance_major_tail: 914, 917.
after: 147, 866, 1196.
after_assignment: 208, 265, 266, 1268.
`\afterassignment` primitive: 265.
after_group: 208, 265, 266, 1271.

\aftergroup primitive: 265.
after_math: 1193, 1194.
after_token: 1266, 1267, 1268, 1269.
aire: 560, 561, 563, 576.
align_error: 1126, 1127.
align_group: 269, 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.
align_state: 88, 309, 324, 325, 331, 339, 342, 347, 357, 394, 395, 396, 403, 442, 475, 482, 483, 486, 770, 771, 772, 774, 777, 783, 784, 785, 788, 789, 791, 1069, 1094, 1126, 1127.
aligning: 305, 306, 339, 777, 789.
alignment of rules with characters: 589.
alpha: 560, 571, 572.
alpha_file: 25,* 27,* 28,* 31,* 50, 54, 304, 480, 525, 1342.
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.
Amble, Ole: 925.
AmSTeX: 1331.
any_mode: 1045, 1048, 1057, 1063, 1067, 1073, 1097, 1102, 1104, 1126, 1134, 1210, 1268, 1271, 1274, 1276, 1285, 1290, 1347.
any_state_plus: 344, 345, 347.
app_lc_hex: 48.
app_space: 1030, 1043.
append_char: 42, 48, 52, 58, 180, 195, 260, 516,* 525, 692, 695, 939.
append_charnode_to_t: 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_vlist: 679, 799, 888, 1076, 1203, 1204, 1205.
area_delimiter: 513, 515, 516,* 517.
arg: 36.*
argc: 36.*
Argument of \x has...: 395.
arith_error: 104, 105, 106, 107, 448, 453, 460, 1236.
Arithmetic overflow: 1236.
artificial_demerits: 830, 851, 854, 855, 856.
ASCII code: 17, 503.
ASCII_code: 18, 19, 20, 29, 30, 31,* 38, 42, 54, 58, 60, 82, 292, 341, 389, 516,* 519, 523, 692, 892, 912, 921, 943, 950, 953, 959, 960, 1376.
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, 239, 413, 1210, 1222, 1224, 1228, 1237.
assign_mu_glue: 209, 226, 227, 413, 1210, 1222, 1224, 1228, 1237.
assign_toks: 209, 230, 231, 233, 323, 413, 415, 1210, 1224, 1226, 1227.
at: 1258.
\atop primitive: 1178.
atop_code: 1178, 1179, 1182.
\atopwithdelims primitive: 1178.
attach_fraction: 448, 453, 454, 456.
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, 1311, 1312.
AVAIL list clobbered...: 168.
awful_bad: 833, 834, 835, 836, 854, 874, 970, 974, 975, 987, 1005, 1006, 1007.
axis_height: 700, 706, 736, 746, 747, 749, 762.
b: 464, 465, 470, 498, 523, 560, 597,* 679, 705, 706, 709, 711, 715, 830, 970, 994, 1198, 1247, 1288.
b_close: 28,* 560.
b_make_name_string: 525.
b_open_in: 27,* 563.
b_open_out: 27,* 532.*
back_error: 327, 373, 396, 403, 415, 442, 446, 476, 479, 503, 577, 783, 1078, 1084, 1161, 1197, 1207, 1212.
back_input: 281, 325, 326, 327, 368, 369, 372, 375, 379, 395, 405, 407, 415, 443, 444, 448, 452, 455, 461, 526, 788, 1031, 1047, 1054, 1064, 1090, 1095, 1124, 1127, 1132, 1138, 1150, 1152, 1153, 1215, 1221, 1226, 1269, 1375.
back_list: 323, 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: 961.
Bad \prevgraf: 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_fmt: 1303, 1306, 1308, 1312, 1317, 1327.
bad_pool: 51, 52, 53.
bad_tfm: 560.
badness: 108, 660, 667, 674, 678, 828, 852, 853, 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.
baseline_skip: 224, 247, 679.
\baselineskip primitive: 226.
baseline_skip_code: 149, 224, 225, 226, 679.
batch_mode: 73, 75, 86, 90, 92, 93, 535, 1262, 1263, 1327, 1328.
\batchmode primitive: 1262.
bc: 540, 541, 543, 545, 560, 565, 566, 570, 576.
bch_label: 560, 573, 576.
bchar: 560, 573, 576, 901, 903, 905, 906, 908, 911, 913, 916, 917, 1032, 1034, 1037, 1038, 1040.
bchar_label: 549, 552, 576, 909, 916, 1034, 1040, 1322, 1323.
before: 147, 192, 1196.
begin: 7* 8.
begin_box: 1073, 1079, 1084.
begin_diagnostic: 76, 245, 284, 299, 323, 400, 401, 502, 509, 581, 638, 641, 663, 675, 863, 987, 992, 1006, 1011, 1121, 1293, 1296.
begin_file_reading: 78, 87, 328, 483, 537*
begin_group: 208, 265, 266, 1063.
\begingroup primitive: 265.
begin_insert_or_adjust: 1097, 1099.
begin_name: 512, 515, 526, 527, 531.
begin_pseudoprint: 316, 318, 319.
begin_token_list: 323, 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_skip: 224.
\belowdisplayskip primitive: 226.
below_display_skip_code: 224, 225, 226, 1203, 1206.
best_bet: 872, 874, 875, 877* 878.
best_height_plus_depth: 971, 974, 1010, 1011.
best_ins_ptr: 981, 1005, 1009, 1018, 1020, 1021.
best_line: 872, 874, 875, 877* 890.
best_page_break: 980, 1005, 1013, 1014.
best_pl_line: 833, 845, 855.
best_place: 833, 845, 855, 970, 974, 980.
best_size: 980, 1005, 1017.
beta: 560, 571, 572.
big_op_spacing1: 701, 751.
big_op_spacing2: 701, 751.
big_op_spacing3: 701, 751.
big_op_spacing4: 701, 751.
big_op_spacing5: 701, 751.
big_switch: 209, 236, 994, 1029, 1030, 1031, 1036, 1041.
BigEndian order: 540.
billion: 625.
bin_noad: 682, 690, 696, 698, 728, 729, 761, 1156, 1157.
bin_op_penalty: 236, 761.
\binoppenalty primitive: 238.
bin_op_penalty_code: 236, 237, 238.
blank_line: 245.
boolean: 27* 31* 37* 45, 46, 47, 76, 79* 96* 104, 106, 107, 165, 167, 245, 256, 311, 361, 407, 413, 440, 448, 461, 473, 498, 516* 524, 527, 549, 560, 578, 592, 619, 629, 645, 706, 719, 726, 791, 825, 828, 829, 830, 862* 877* 900, 907, 950, 960, 989, 1012, 1032, 1051, 1054, 1091, 1160, 1194, 1211, 1281, 1303, 1342.
bop: 583, 585, 586, 588, 590, 592, 638, 640.
Bosshard, Hans Rudolf: 458.
bot: 546.
bot_mark: 382, 383, 1012, 1016.
\botmark primitive: 384.
bot_mark_code: 382, 384, 385.
bottom_level: 269, 272, 281, 1064, 1068.
bottom_line: 311.
bowels: 592.
box: 230, 232, 420, 505, 977, 992, 993, 1009, 1015, 1017, 1018, 1021, 1023, 1028, 1079, 1110, 1247, 1296.
\box primitive: 1071.
box_base: 230, 232, 233, 255, 1077.
box_code: 1071, 1072, 1079, 1107, 1110.

box_context: 1075, 1076, 1077, 1078, 1079, 1083, 1084.
box_end: 1075, 1079, 1084, 1086.
box_error: 992, 993, 1015, 1028.
box_flag: 1071, 1075, 1077, 1083, 1241.
box_max_depth: 247, 1086.
`\boxmaxdepth` primitive: 248.
box_max_depth_code: 247, 248.
box_node_size: 135, 136, 202, 206, 649, 668, 715, 727, 751, 756, 977, 1021, 1100, 1110, 1201.
box_ref: 210, 232, 275, 1077.
box_there: 980, 987, 1000, 1001.
`\box255 is not void`: 1015.
`bp`: 458.
brain: 1029.
breadth_max: 181, 182, 198, 233, 236, 1339*
break_node: 819, 845, 855, 856, 864, 877*, 878.
break_penalty: 208, 265, 266, 1102.
break_type: 829, 837, 845, 846, 859.
break_width: 823, 824, 837, 838, 840, 841, 842, 843, 844, 879*
breakpoint: 1338*
broken_ins: 981, 986, 1010, 1021.
broken_penalty: 236, 890.
`\brokenpenalty` primitive: 238.
broken_penalty_code: 236, 237, 238.
broken_ptr: 981, 1010, 1021.
buf_size: 11*, 30, 31*, 35, 36*, 71, 111, 315, 328, 331, 341, 363, 366, 374, 524, 530, 534, 1334.
buffer: 30, 31*, 36*, 37*, 45, 71, 83, 87, 88, 259, 260, 261, 264, 302, 303, 315, 318, 331, 341, 343, 352, 354, 355, 356, 360*, 362, 363, 366, 374, 483, 484, 523, 524, 530, 531, 534, 538, 597*, 1337, 1339*
`Buffer size exceeded`: 35.
build_choices: 1173, 1174.
build_discretionary: 1118, 1119.
build_page: 800, 812, 988, 994, 1026, 1054, 1060, 1076, 1091, 1094, 1100, 1103, 1145, 1200.
`by`: 1236.
bypass_eoln: 31*
byte: 25*
byte_file: 25*, 27*, 28*, 525, 539.
b0: 110, 113, 114, 133, 221, 268, 545, 546, 550, 554, 556, 564, 602, 683, 685, 921, 958, 1309, 1310.
b1: 110, 113, 114, 133, 221, 268, 545, 546, 554, 556, 564, 602, 683, 685, 921, 958, 1309, 1310.
b2: 110, 113, 114, 545, 546, 554, 556, 564, 602, 683, 685, 1309, 1310.
b3: 110, 113, 114, 545, 546, 556, 564, 602, 683, 685, 1309, 1310.
c: 47, 63, 82, 144, 264, 274, 292, 341, 470, 516*, 519, 523, 560, 581, 582, 592, 645, 692, 694, 706, 709, 711, 712, 738, 749, 893, 912, 953, 959, 960, 994, 1012, 1086, 1110, 1117, 1136, 1151, 1155, 1181, 1243, 1245, 1246, 1247, 1275, 1279, 1288, 1335.
c_leaders: 149, 190, 627, 636, 1071, 1072.
`\cleaders` primitive: 1071.
c_loc: 912, 916.
call: 210, 223, 275, 296, 366, 380, 387, 395, 396, 507, 1218, 1221, 1225, 1226, 1227, 1295.
cancel_boundary: 1030, 1032, 1033, 1034.
`cannot \read`: 484.
car_ret: 207, 232, 342, 347, 777, 780, 781, 783, 784, 785, 788, 1126.
carriage_return: 22, 49, 207, 232, 240, 363.
case_shift: 208, 1285, 1286, 1287.
cat: 341, 354, 355, 356.
cat_code: 230, 232, 236, 262, 341, 343, 354, 355, 356, 1337.
`\catcode` primitive: 1230.
cat_code_base: 230, 232, 233, 235, 1230, 1231, 1233.
cc: 36*, 341, 352, 355.
`cc`: 458.
change_if_limit: 497, 498, 509.
char: 19, 26, 36*, 520, 534.
`\char` primitive: 265.
char_base: 550, 552, 554, 566, 570, 576, 1322, 1323.
char_box: 709, 710, 711, 738.
`\chardef` primitive: 1222.
char_def_code: 1222, 1223, 1224.
char_depth: 554, 654, 708, 709, 712.
char_depth_end: 554.
char_exists: 554, 573, 576, 582, 708, 722, 738, 740, 749, 755, 1036.
char_given: 208, 413, 935, 1030, 1038, 1090, 1124, 1151, 1154, 1222, 1223, 1224.
char_height: 554, 654, 708, 709, 712, 1125.
char_height_end: 554.
char_info: 543, 550, 554, 555, 557, 570, 573, 576, 582, 620, 654, 708, 709, 712, 714, 715, 722, 724, 738, 740, 749, 841, 842, 866, 867, 870, 871, 909, 1036, 1037, 1039, 1040, 1113, 1123, 1125, 1147.
char_info_end: 554.
char_info_word: 541, 543, 544.
char_italic: 554, 709, 714, 749, 755, 1113.
char_italic_end: 554.
char_kern: 557, 741, 753, 909, 1040.
char_kern_end: 557.
char_node: 134, 143, 145, 162, 176, 548, 592, 620, 649, 752, 881, 907, 1029, 1113, 1138.
char_num: 208, 265, 266, 935, 1030, 1038, 1090, 1124, 1151, 1154.
char_tag: 554, 570, 708, 710, 740, 741, 749, 752, 909, 1039.

char_warning: 581, 582, 722, 1036.
char_width: 554, 620, 654, 709, 714, 715, 740, 841, 842, 866, 867, 870, 871, 1123, 1125, 1147.
char_width_end: 554.
character: 134, 143, 144, 174, 176, 206, 582, 620, 654, 681, 682, 683, 687, 691, 709, 715, 722, 724, 749, 752, 753, 841, 842, 866, 867, 870, 871, 896, 897, 898, 903, 907, 908, 910, 911, 1032, 1034, 1035, 1036, 1037, 1038, 1040, 1113, 1123, 1125, 1147, 1151, 1155, 1165.
character set dependencies: 23, 49.
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, 1339*
check_outer_validity: 336, 351, 353, 354, 357, 362, 375.
check_shrinkage: 825, 827, 868.
Chinese characters: 134, 585.
choice_node: 688, 689, 690, 698, 730.
choose_mlist: 731.
chr: 19, 20, 23, 24, 1222.
chr_cmd: 298, 781.
chr_code: 227, 231, 239, 249, 298, 377, 385, 411, 412, 413, 417, 469, 488, 492, 781, 984, 1053, 1059, 1071, 1072, 1089, 1108, 1115, 1143, 1157, 1170, 1179, 1189, 1209, 1220, 1223, 1231, 1251, 1255, 1261, 1263, 1273, 1278, 1287, 1289, 1292, 1346.
cinteger: 1381*
clang: 212, 213, 812, 1034, 1091, 1200, 1376, 1377.
clean_box: 720, 734, 735, 737, 738, 742, 744, 749, 750, 757, 758, 759.
clear_for_error_prompt: 78, 83, 330, 346.
clear_io_result: 27*
clear_terminal: 34;* 330, 530.
CLOBBERED: 293.
clobbered: 167, 168, 169.
close: 28*
close_files_and_terminate: 78, 81, 1332;* 1333*
\closein primitive: 1272.
close_noad: 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.
clr: 737, 743, 745, 746, 756, 757, 758, 759.
club_penalty: 236, 890.
\clubpenalty primitive: 238.
club_penalty_code: 236, 237, 238.
cm: 458.
cmd: 298, 1222, 1289.
cmd_line: 1380*
co_backup: 366.
combine_two_deltas: 860.
comment: 207, 232, 347.
common_ending: 15, 498, 500, 509, 649, 660, 666, 667, 668, 674, 677, 678, 895, 903, 1257, 1260, 1293, 1294, 1297.
Completed box...: 638.
compress_trie: 949, 952.
cond_math_glue: 149, 189, 732, 1171.
cond_ptr: 489, 490, 495, 496, 497, 498, 500, 509, 1335.
conditional: 366, 367, 498.
confusion: 95, 202, 206, 281, 497, 630, 669, 728, 736, 754, 761, 766, 791, 798, 800, 841, 842, 866, 870, 871, 877;* 968, 973, 1000, 1068, 1185, 1200, 1211, 1348, 1357, 1358, 1373.
continental_point_token: 438, 448.
continue: 15, 82, 83, 84;* 88, 89, 389, 392, 393, 394, 395, 397, 706, 708, 774, 784, 815, 829, 832, 851, 896, 906, 909, 910, 911, 994, 1001.
contrib_head: 162, 215, 218, 988, 994, 995, 998, 999, 1001, 1017, 1023, 1026.
contrib_tail: 995, 1017, 1023, 1026.
contribute: 994, 997, 1000, 1002, 1008, 1364.
conv_toks: 366, 367, 470.
conventions for representing stacks: 300.
convert: 210, 366, 367, 468, 469, 470.
convert_to_break_width: 843.
\copy primitive: 1071.
copy_code: 1071, 1072, 1079, 1107, 1108, 1110.
copy_node_list: 161, 203, 204, 206, 1079, 1110.
copy_to_cur_active: 829, 861.
count: 236, 427, 638, 640, 986, 1008, 1009, 1010.
\count primitive: 411.
count_base: 236, 239, 242, 1224, 1237.
\countdef primitive: 1222.
count_def_code: 1222, 1223, 1224.
\cr primitive: 780.
cr_code: 780, 781, 789, 791, 792.
\crcr primitive: 780.
cr_cr_code: 780, 785, 789.
cramped: 688, 702.
cramped_style: 702, 734, 737, 738.
cs_count: 256, 258, 260, 1318, 1319, 1334.
cs_error: 1134, 1135.
cs_name: 210, 265, 266, 366, 367.
\csname primitive: 265.

cs_token_flag: 289, 290, 293, 334, 336, 337, 339, 357, 358, 365, 369, 372, 375, 379, 380, 381, 442, 466, 506, 780, 1065, 1132, 1215, 1289, 1314, 1371.
cur_active_width: 823, 824, 829, 832, 837, 843, 844, 851, 852, 853, 860.
cur_align: 770, 771, 772, 777, 778, 779, 783, 786, 788, 789, 791, 792, 795, 796, 798.
cur_area: 512, 517, 529, 530, 537* 1257, 1260, 1351, 1374.
cur_boundary: 270, 271, 272, 274, 282.
cur_box: 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1084, 1086, 1087.
cur_break: 821, 845, 879* 880, 881.
cur_c: 722, 723, 724, 738, 749, 752, 753, 755.
cur_chr: 88, 296, 297, 299, 332, 337, 341, 343, 348, 349, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360* 364, 365, 378, 380, 381, 386, 387, 389, 403, 407, 413, 424, 428, 442, 470, 472, 474, 476, 479, 483, 494, 495, 498, 500, 506, 507, 508, 509, 510, 526, 577, 782, 785, 789, 935, 937, 962, 1030, 1034, 1036, 1038, 1049, 1058, 1060, 1061, 1066, 1073, 1079, 1083, 1090, 1093, 1105, 1106, 1110, 1117, 1124, 1128, 1140, 1142, 1151, 1152, 1154, 1155, 1158, 1159, 1160, 1171, 1181, 1191, 1211, 1212, 1213, 1217, 1218, 1221, 1224, 1226, 1227, 1228, 1232, 1233, 1234, 1237, 1243, 1245, 1246, 1247, 1252, 1253, 1265, 1275, 1279, 1288, 1293, 1335, 1348, 1350, 1375.
cur_cmd: 88, 211, 296, 297, 299, 332, 337, 341, 342, 343, 344, 348, 349, 351, 353, 354, 357, 358, 360* 364, 365, 366, 367, 368, 372, 380, 381, 386, 387, 403, 404, 406, 407, 413, 415, 428, 440, 442, 443, 444, 448, 452, 455, 461, 463, 474, 476, 477, 478, 479, 483, 494, 506, 507, 526, 577, 777, 782, 783, 784, 785, 788, 789, 935, 961, 1029, 1030, 1038, 1049, 1066, 1078, 1079, 1084, 1095, 1099, 1124, 1128, 1138, 1151, 1152, 1160, 1165, 1176, 1177, 1197, 1206, 1211, 1212, 1213, 1221, 1226, 1227, 1228, 1236, 1237, 1252, 1270, 1375.
cur_cs: 297, 332, 333, 336, 337, 338, 341, 351, 353, 354, 356, 357, 358, 365, 372, 374, 379, 380, 381, 389, 391, 407, 472, 473, 507, 774, 1152, 1215, 1218, 1221, 1224, 1225, 1226, 1257, 1294, 1352, 1371.
cur_ext: 512, 517, 529, 530, 537* 1275, 1351, 1374.
cur_f: 722, 724, 738, 741, 749, 752, 753, 755.
cur_fam: 236, 1151, 1155, 1165.
cur_fam_code: 236, 237, 238, 1139, 1145.
cur_file: 304, 329, 362, 537* 538.
cur_font: 230, 232, 558, 559, 577, 1032, 1034, 1042, 1044, 1117, 1123, 1124, 1146.
cur_font_loc: 230, 232, 233, 234, 1217.
cur_g: 619, 625, 629, 634.
cur_glue: 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, 534, 1131.
cur_l: 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_length: 41, 180, 182, 260, 516* 525, 617, 692, 1368.
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_loop: 770, 771, 772, 777, 783, 792, 793, 794.
cur_mark: 296, 382, 386, 1335.
cur_mlist: 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_p: 823, 828, 829, 830, 833, 837, 839, 840, 845, 851, 853, 855, 856, 857, 858, 859, 860, 862* 863, 865, 866, 867, 868, 869, 872, 877* 878, 879* 880, 881, 894, 903, 1362.
cur_q: 907, 908, 910, 911, 1034, 1035, 1036, 1037, 1040.
cur_r: 907, 908, 909, 910, 911, 1032, 1034, 1037, 1038, 1039, 1040.
cur_rh: 906, 908, 909, 910.
cur_s: 593, 616, 619, 629, 640, 642*
cur_size: 700, 701, 703, 719, 722, 723, 732, 736, 737, 744, 746, 747, 748, 749, 757, 758, 759, 762.
cur_span: 770, 771, 772, 787, 796, 798.
cur_style: 703, 719, 720, 726, 730, 731, 734, 735, 737, 738, 742, 744, 745, 746, 748, 749, 750, 754, 756, 757, 758, 759, 760, 763, 766, 1194, 1196, 1199.
cur_tail: 770, 771, 772, 786, 796, 799.
cur_tok: 88, 281, 297, 325, 326, 327, 336, 364, 365, 366, 368, 369, 372, 375, 379, 380, 381,

392, 393, 394, 395, 397, 399, 403, 405, 407, 440, 441, 442, 444, 445, 448, 452, 474, 476, 477, 479, 483, 494, 503, 506, 783, 784, 1038, 1047, 1095, 1127, 1128, 1132, 1215, 1221, 1268, 1269, 1271, 1371, 1372.
cur_v: 616, 618, 619, 623, 624, 628, 629, 631, 632, 633, 635, 636, 637, 640.
cur_val: 264, 265, 334, 366, 410, 413, 414, 415, 419, 420, 421, 423, 424, 425, 426, 427, 429, 430, 431, 433, 434, 435, 436, 437, 438, 439, 440, 442, 444, 445, 447, 448, 450, 451, 453, 455, 457, 458, 460, 461, 462, 463, 465, 466, 472, 482, 491, 501, 503, 504, 505, 509, 553, 577, 578, 579, 580, 645, 780, 782, 935, 1030, 1038, 1060, 1061, 1073, 1079, 1082, 1099, 1103, 1110, 1123, 1124, 1151, 1154, 1160, 1161, 1165, 1182, 1188, 1224, 1225, 1226, 1227, 1228, 1229, 1232, 1234, 1236, 1237, 1238, 1239, 1240, 1241, 1243, 1244, 1245, 1246, 1247, 1248, 1253, 1258, 1259, 1275, 1296, 1344, 1350, 1377.
cur_val_level: 366, 410, 413, 419, 420, 421, 423, 424, 427, 429, 430, 439, 449, 451, 455, 461, 465, 466.
cur_width: 877;* 889.
current page: 980.
current_character_being_worked_on: 570.
cv_backup: 366.
cvl_backup: 366.
d: 107, 176, 177, 259, 341, 440, 560, 649, 668, 679, 706, 830, 944, 970, 1068, 1086, 1138, 1198.
d_fixed: 608, 609.
danger: 1194, 1195, 1199.
data: 210, 232, 1217, 1232, 1234.
data structure assumptions: 161, 164, 204, 816;* 968, 981, 1289.
day: 236, 241;* 536, 617, 1328.
`\day` primitive: 238.
day_code: 236, 237, 238.
`dd`: 458.
deactivate: 829, 851, 854.
dead_cycles: 419, 592, 593, 638, 1012, 1024, 1025, 1054, 1242, 1246.
`\deadcycles` primitive: 416.
debug: 7;* 9;* 78, 84;* 93, 114, 165, 166, 167, 172, 1031, 1338.*
`debug #`: 1338.*
debug_help: 78, 84;* 93, 1338.*
debugging: 7;* 84;* 96;* 114, 165, 182, 1031, 1338.*
decent_fit: 817, 834, 852, 853, 864.
decr: 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;* 362, 394, 399, 422, 429, 442, 477, 483, 494, 509, 534, 538, 568, 576, 601, 619, 629, 638, 642;* 643, 716, 717, 803, 808, 840, 858, 869, 883, 915, 916, 930, 931, 940, 944, 948, 965, 1060, 1100, 1120, 1127, 1131, 1174, 1186, 1194, 1244, 1293, 1311, 1335, 1337.
def: 209, 1208, 1209, 1210, 1213, 1218.
`\def` primitive: 1208.
def_code: 209, 413, 1210, 1230, 1231, 1232.
def_family: 209, 413, 577, 1210, 1230, 1231, 1234.
def_font: 209, 265, 266, 413, 577, 1210, 1256.
def_ref: 305, 306, 473, 482, 960, 1101, 1218, 1226, 1279, 1288, 1352, 1354, 1370.
default_code: 683, 697, 743, 1182.
default_hyphen_char: 236, 576.
`\defaulthyphenchar` primitive: 238.
default_hyphen_char_code: 236, 237, 238.
default_rule: 463.
default_rule_thickness: 683, 701, 734, 735, 737, 743, 745, 759.
default_skew_char: 236, 576.
`\defaultskewchar` primitive: 238.
default_skew_char_code: 236, 237, 238.
defecation: 597.*
define: 1214, 1217, 1218, 1221, 1224, 1225, 1226, 1227, 1228, 1232, 1234, 1236, 1248, 1257.
defining: 305, 306, 339, 473, 482.
del_code: 236, 240, 1160.
`\delcode` primitive: 1230.
del_code_base: 236, 240, 242, 1230, 1232, 1233.
delete_glue_ref: 201, 202, 275, 451, 465, 578, 732, 802, 816;* 826, 881, 976, 996, 1004, 1017, 1022, 1100, 1229, 1236, 1239, 1335.
delete_last: 1104, 1105.
delete_q: 726, 760, 763.
delete_token_ref: 200, 202, 275, 324, 977, 979, 1012, 1016, 1335, 1358.
deletions_allowed: 76, 77, 84;* 85, 98, 336, 346.
delim_num: 207, 265, 266, 1046, 1151, 1154, 1160.
delimited_code: 1178, 1179, 1182, 1183.
delimiter: 687, 696, 762, 1191.
`\delimiter` primitive: 265.
delimiter_factor: 236, 762.
`\delimiterfactor` primitive: 238.
delimiter_factor_code: 236, 237, 238.
delimiter_shortfall: 247, 762.
`\delimitershortfall` primitive: 248.
delimiter_shortfall_code: 247, 248.
delim1: 700, 748.
delim2: 700, 748.
delta: 103, 726, 728, 733, 735, 736, 737, 738, 742, 743, 745, 746, 747, 748, 749, 750, 754, 755, 756,

759, 762, 994, 1008, 1010, 1123, 1125.
delta_node: 822, 830, 832, 843, 844, 860, 861, 865, 874, 875.
delta_node_size: 822, 843, 844, 860, 861, 865.
delta1: 743, 746, 762.
delta2: 743, 746, 762.
den: 585, 587, 590.
denom: 450, 458.
denom_style: 702, 744.
denominator: 683, 690, 697, 698, 744, 1181, 1185.
denom1: 700, 744.
denom2: 700, 744.
deplorable: 974, 1005.
`depth`: 463.
depth: 135, 136, 138, 139, 140, 184, 187, 188, 463, 554, 622, 624, 626, 631, 632, 635, 641, 649, 653, 656, 668, 670, 679, 688, 704, 706, 709, 713, 727, 730, 731, 735, 736, 737, 745, 746, 747, 749, 750, 751, 756, 758, 759, 768, 769, 801, 806, 810, 973, 1002, 1009, 1010, 1021, 1087, 1100.
depth_base: 550, 552, 554, 566, 571, 1322, 1323.
depth_index: 543, 554.
depth_offset: 135, 416, 769, 1247.
depth_threshold: 181, 182, 198, 233, 236, 692, 1339*
dig: 54, 64, 65, 67, 102, 452.
digit_sensed: 960, 961, 962.
dimen: 247, 427, 1008, 1010.
`\dimen` primitive: 411.
dimen_base: 220, 236, 247, 248, 249, 250, 251, 252, 1070, 1145.
`\dimendef` primitive: 1222.
dimen_def_code: 1222, 1223, 1224.
dimen_par: 247.
dimen_pars: 247.
dimen_val: 410, 411, 412, 413, 415, 416, 417, 418, 420, 421, 424, 425, 427, 428, 429, 449, 455, 465, 1237.
`Dimension too large`: 460.
dirty Pascal: 3, 114, 172, 182, 186, 285, 812, 1331.
disc_break: 877,* 880, 881, 882, 890.
disc_group: 269, 1117, 1118, 1119.
disc_node: 145, 148, 175, 183, 202, 206, 730, 761, 817, 819, 829, 856, 858, 866, 881, 914, 1081, 1105.
disc_width: 839, 840, 869, 870.
discretionary: 208, 1090, 1114, 1115, 1116.
`Discretionary list is too long`: 1120.
`\discretionary` primitive: 1114.
`Display math...with $$`: 1197.
display_indent: 247, 800, 1138, 1145, 1199.
`\displayindent` primitive: 248.
display_indent_code: 247, 248, 1145.
`\displaylimits` primitive: 1156.
display_mlist: 689, 695, 698, 731, 1174.
display_style: 688, 694, 731, 1169, 1199.
`\displaystyle` primitive: 1169.
display_widow_penalty: 236, 1145.
`\displaywidowpenalty` primitive: 238.
display_widow_penalty_code: 236, 237, 238.
display_width: 247, 1138, 1145, 1199.
`\displaywidth` primitive: 248.
display_width_code: 247, 248, 1145.
div: 100, 627, 636.
divide: 209, 265, 266, 1210, 1235, 1236.
`\divide` primitive: 265.
do_all_six: 823, 829, 832, 837, 843, 844, 860, 861, 864, 970, 987.
do_assignments: 800, 1123, 1206, 1270.
do_endv: 1130, 1131.
do_extension: 1347, 1348, 1375.
do_nothing: 16, 27,* 33,* 34,* 57, 58, 84,* 175, 202, 275, 344, 357, 538, 569, 609, 611, 612, 622, 631, 651, 669, 692, 728, 733, 761, 837, 866, 899, 1045, 1236, 1359, 1360, 1373, 1382*
do_register_command: 1235, 1236.
doing_leaders: 592, 593, 628, 637, 1374.
done: 15, 47, 53, 202, 281, 282, 311, 380, 389, 397, 440, 445, 448, 453, 458, 473, 474, 476, 482, 483, 494, 526, 530, 531, 537,* 560, 567, 576, 615, 638, 640, 641, 698, 726, 738, 740, 760, 761, 774, 777, 815, 829, 837, 863, 873, 877,* 881, 895, 906, 909, 911, 931, 960, 961, 970, 974, 977, 979, 994, 997, 998, 1005, 1079, 1081, 1119, 1121, 1138, 1146, 1211, 1227, 1252, 1358.
done_with_noad: 726, 727, 728, 733, 754.
done_with_node: 726, 727, 730, 731, 754.
done1: 15, 167, 168, 389, 399, 448, 452, 473, 474, 738, 741, 774, 783, 815, 829, 852, 877,* 879,* 894, 896, 899, 960, 965, 994, 997, 1000, 1302, 1315.
done2: 15, 167, 169, 448, 458, 459, 473, 478, 774, 784, 815, 896, 1302, 1316.
done3: 15, 815, 897, 898.
done4: 15, 815, 899.
done5: 15, 815, 866, 869.
done6: 15.
dont_expand: 210, 258, 357, 369.
`Double subscript`: 1177.
`Double superscript`: 1177.
double_hyphen_demerits: 236, 859.
`\doublehyphendemerits` primitive: 238.
double_hyphen_demerits_code: 236, 237, 238.
`Doubly free location...`: 169.
down_ptr: 605, 606, 607, 615.
downdate_width: 860.

down1: 585, 586, 607, 609, 610, 613, 614, 616.
down2: 585, 594, 610.
down3: 585, 610.
down4: 585, 610.
\dp primitive: 416.
dry rot: 95.
\dump...only by INITEX: 1335.
\dump primitive: 1052.
dump_four_ASCII: 1309.
dump_hh: 1305, 1318, 1324.
dump_int: 1305, 1307, 1309, 1311, 1313, 1315, 1316, 1318, 1320, 1322, 1324, 1326.
dump_qqqq: 1305, 1309, 1322.
dump_wd: 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*, 592, 595, 597*, 642*.
DVI files: 583.
dvi_font_def: 602, 621, 643.
dvi_four: 600, 602, 610, 617, 624, 633, 640, 642*, 1368.
dvi_gone: 594, 595, 596, 598, 612.
dvi_h: 616, 617, 619, 620, 623, 624, 628, 629, 632, 637.
dvi_index: 594, 595, 597*.
dvi_limit: 594, 595, 596, 598, 599.
dvi_offset: 594, 595, 596, 598, 601, 605, 607, 613, 614, 619, 629, 640, 642*.
dvi_out: 598, 600, 601, 602, 603, 609, 610, 617, 619, 620, 621, 624, 629, 633, 640, 642*, 1368.
dvi_pop: 601, 619, 629.
dvi_ptr: 594, 595, 596, 598, 599, 601, 607, 619, 629, 640, 642*.
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, 560, 565, 566, 570, 576.
\edef primitive: 1208.
edge: 619, 623, 626, 629, 635.
edit_file_name: 79*, 80*, 84*, 1332*, 1380*.
edit_line: 79*, 80*, 84*, 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, 498.
em: 455.
Emergency stop: 93.
emergency_stretch: 247, 828, 863.
\emergencystretch primitive: 248.
emergency_stretch_code: 247, 248.
empty: 16, 421, 681, 685, 687, 692, 722, 723, 738, 749, 751, 752, 754, 755, 756, 980, 986, 987, 991, 1001, 1008, 1176, 1177, 1186.
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*.
End of file on the terminal: 37*, 71.
(\end occurred...): 1335.
\end primitive: 1052.
end_cs_name: 208, 265, 266, 372, 1134.
\endcsname primitive: 265.
end_diagnostic: 245, 284, 299, 323, 400, 401, 502, 509, 581, 638, 641, 663, 675, 863, 987, 992, 1006, 1011, 1121, 1298.
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: 238.
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.
end_write_token: 1371, 1372.
endcases: 10*.
endv: 207, 298, 375, 380, 768, 780, 782, 791, 1046, 1130, 1131.
ensure_dvi_open: 532*, 617.

ensure_vbox: 993, 1009, 1018.
eof: 26, 31,* 52, 564, 575, 1327, 1338,* 1339.*
eoln: 31,* 52.
eop: 583, 585, 586, 588, 640, 642.*
eq_define: 277, 278, 279, 372, 782, 1070, 1077, 1214.
eq_destroy: 275, 277, 279, 283.
eq_level: 221, 222, 228, 232, 236, 253, 264, 277, 279, 283, 780, 977, 1315, 1369.
eq_level_field: 221.
eq_no: 208, 1140, 1141, 1143, 1144.
`\eqno` primitive: 1141.
eq_save: 276, 277, 278.
eq_type: 210, 221, 222, 223, 228, 232, 253, 258, 264, 265, 267, 277, 279, 351, 353, 354, 357, 358, 372, 389, 391, 780, 1152, 1315, 1369.
eq_type_field: 221, 275.
eq_word_define: 278, 279, 1070, 1139, 1145, 1214.
eqtb: 115, 163, 220, 221, 222, 223, 224, 228, 230, 232, 236, 240, 242, 247, 250, 251, 252, 253, 255, 262, 264, 265, 266, 267, 268, 270, 272, 274, 275, 276, 277, 278, 279, 281, 282, 283, 284, 285, 286, 289, 291, 297, 298, 305, 307, 332, 333, 354, 389, 413, 414, 473, 491, 548, 553, 780, 814, 1188, 1208, 1222, 1238, 1240, 1253, 1257, 1315, 1316, 1317, 1339,* 1345.
eqtb_size: 220, 247, 250, 252, 253, 254, 1307, 1308, 1316, 1317.
equiv: 221, 222, 223, 224, 228, 229, 230, 232, 233, 234, 235, 253, 255, 264, 265, 267, 275, 277, 279, 351, 353, 354, 357, 358, 413, 414, 415, 508, 577, 780, 1152, 1227, 1239, 1240, 1257, 1289, 1315, 1369.
equiv_field: 221, 275, 285.
err_help: 79,* 230, 1283, 1284.
`\errhelp` primitive: 230.
err_help_loc: 230.
`\errmessage` primitive: 1277.
error: 72, 75, 76, 78, 79,* 82, 88, 91, 93, 98, 327, 338, 346, 370, 398, 408, 418, 428, 445, 454, 456, 459, 460, 475, 476, 486, 500, 510, 523, 535, 561, 567, 579, 641, 723, 776, 784, 792, 826, 936, 937, 960, 961, 962, 963, 976, 978, 992, 1004, 1009, 1024, 1027, 1050, 1064, 1066, 1068, 1069, 1080, 1082, 1095, 1099, 1106, 1110, 1120, 1121, 1128, 1129, 1135, 1159, 1166, 1177, 1183, 1192, 1195, 1213, 1225, 1232, 1236, 1237, 1241, 1252, 1259, 1283, 1284, 1293, 1372, 1382.*
error_context_lines: 236, 311.
`\errorcontextlines` primitive: 238.
error_context_lines_code: 236, 237, 238.
error_count: 76, 77, 82, 86, 1096, 1293.
error_line: 11,* 14, 54, 58, 306, 311, 315, 316, 317.
error_message_issued: 76, 82, 95.
error_stop_mode: 72, 73, 74, 82, 93, 98, 1262, 1283, 1293, 1294, 1297, 1327, 1335.
`\errorstopmode` primitive: 1262.
erstat: 27.*
escape: 207, 232, 344, 1337.
escape_char: 236, 240, 243.
`\escapechar` primitive: 238.
escape_char_code: 236, 237, 238.
`etc`: 182.
`ETC`: 292.
every_cr: 230, 774, 799.
`\everycr` primitive: 230.
every_cr_loc: 230, 231.
every_cr_text: 307, 314, 774, 799.
every_display: 230, 1145.
`\everydisplay` primitive: 230.
every_display_loc: 230, 231.
every_display_text: 307, 314, 1145.
every_hbox: 230, 1083.
`\everyhbox` primitive: 230.
every_hbox_loc: 230, 231.
every_hbox_text: 307, 314, 1083.
every_job: 230, 1030.
`\everyjob` primitive: 230.
every_job_loc: 230, 231.
every_job_text: 307, 314, 1030.
every_math: 230, 1139.
`\everymath` primitive: 230.
every_math_loc: 230, 231.
every_math_text: 307, 314, 1139.
every_par: 230, 1091.
`\everypar` primitive: 230.
every_par_loc: 230, 231, 307, 1226.
every_par_text: 307, 314, 1091.
every_vbox: 230, 1083, 1167.
`\everyvbox` primitive: 230.
every_vbox_loc: 230, 231.
every_vbox_text: 307, 314, 1083, 1167.
`ex`: 455.
ex_hyphen_penalty: 145, 236, 869.
`\exhyphenpenalty` primitive: 238.
ex_hyphen_penalty_code: 236, 237, 238.
ex_space: 208, 265, 266, 1030, 1090.
exactly: 644, 645, 715, 889, 977, 1017, 1062, 1201.
exit: 15, 16, 37,* 47, 58, 59, 69, 82, 125, 182, 292, 341, 389, 407, 461, 497, 498, 524, 582, 607, 615, 649, 668, 752, 791, 829, 895, 934, 944, 948, 977, 994, 1012, 1030, 1054, 1079, 1105, 1110, 1113, 1119, 1151, 1159, 1174, 1211, 1236, 1270, 1303, 1335, 1338.*

expand: 358, 366, 368, 371, 380, 381, 439, 467, 478, 498, 510, 782.
expand_after: 210, 265, 266, 366, 367.
`\expandafter` primitive: 265.
explicit: 155, 717, 837, 866, 868, 879,* 1058, 1113.
ext_bot: 546, 713, 714.
ext_delimiter: 513, 515, 516,* 517.
ext_mid: 546, 713, 714.
ext_rep: 546, 713, 714.
ext_tag: 544, 569, 708, 710.
ext_top: 546, 713, 714.
exten: 544.
exten_base: 550, 552, 566, 573, 574, 576, 713, 1322, 1323.
extensible_recipe: 541, 546.
extension: 208, 1344, 1346, 1347, 1375.
extensions to TEX: 2,* 146, 1340.
`Extra \else`: 510.
`Extra \endcsname`: 1135.
`Extra \fi`: 510.
`Extra \or`: 500, 510.
`Extra \right.`: 1192.
`Extra }, or forgotten x`: 1069.
`Extra alignment tab...`: 792.
`Extra x`: 1066.
extra_info: 769, 788, 789, 791, 792.
extra_right_brace: 1068, 1069.
extra_space: 547, 558, 1044.
extra_space_code: 547, 558.
eyes and mouth: 332.
f: 27,* 28,* 31,* 144, 448, 525, 560, 577, 578, 581, 582, 592, 602, 649, 706, 709, 711, 712, 715, 716, 717, 738, 830, 862,* 1068, 1113, 1123, 1138, 1211, 1257.
false: 31,* 37,* 45, 46, 47, 51, 76, 80,* 88, 89, 98, 106, 107, 166, 167, 168, 169, 264, 284, 299, 311, 323, 327, 331, 336, 346, 361, 362, 365, 374, 400, 401, 407, 425, 440, 441, 445, 447, 448, 449, 455, 460, 461, 462, 465, 485, 501, 502, 505, 507, 509, 512, 516,* 524, 526, 528, 538, 551, 563, 581, 593, 706, 720, 722, 754, 774, 791, 826, 828, 837, 851, 854, 863, 881, 903, 906, 910, 911, 951, 954, 960, 961, 962, 963, 966, 987, 990, 1006, 1011, 1020, 1026, 1031, 1033, 1034, 1035, 1040, 1051, 1054, 1061, 1101, 1167, 1182, 1183, 1191, 1192, 1194, 1199, 1226, 1236, 1258, 1270, 1279, 1282, 1283, 1288, 1303, 1325, 1336, 1342, 1343, 1352, 1354, 1371, 1374.
false_bchar: 1032, 1034, 1038.
fam: 681, 682, 683, 687, 691, 722, 723, 752, 753, 1151, 1155, 1165.
`\fam` primitive: 238.
fam_fnt: 230, 700, 701, 707, 722, 1195.
fam_in_range: 1151, 1155, 1165.
fast_delete_glue_ref: 201, 202.
fast_get_avail: 122, 371, 1034, 1038.
fast_store_new_token: 371, 399, 464, 466.
`Fatal format file error`: 1303.
fatal_error: 71, 93, 324, 360,* 484, 530, 535, 782, 789, 791, 1131.
fatal_error_stop: 76, 77, 82, 93, 1332.*
fbyte: 564, 568, 571, 575.
Ferguson, Michael John: 2.*
fetch: 722, 724, 738, 741, 749, 752, 755.
fewest_demerits: 872, 874, 875.
fget: 564, 565, 568, 571, 575.
`\fi` primitive: 491.
fi_code: 489, 491, 492, 494, 498, 500, 509, 510.
fi_or_else: 210, 366, 367, 489, 491, 492, 494, 510.
`fil`: 454.
fil: 135, 150, 164, 177, 454, 650, 659, 665, 1201.
fil_code: 1058, 1059, 1060.
fil_glue: 162, 164, 1060.
fil_neg_code: 1058, 1060.
fil_neg_glue: 162, 164, 1060.
`File ended while scanning...`: 338.
`File ended within \read`: 486.
file_name_size: 11,* 26, 519, 522, 523, 525, 1380.*
file_offset: 54, 55, 57, 58, 62, 537,* 638, 1280.
file_opened: 560, 561, 563.
fill: 135, 150, 164, 650, 659, 665, 1201.
fill_code: 1058, 1059, 1060.
fill_glue: 162, 164, 1054, 1060.
filll: 135, 150, 177, 454, 650, 659, 665, 1201.
fin_align: 773, 785, 800, 1131.
fin_col: 773, 791, 1131.
fin_mlist: 1174, 1184, 1186, 1191, 1194.
fin_row: 773, 799, 1131.
fin_rule: 619, 622, 626, 629, 631, 635.
final_cleanup: 1332,* 1335.
final_end: 6, 35, 331, 1332,* 1337, 1379.*
final_hyphen_demerits: 236, 859.
`\finalhyphendemerits` primitive: 238.
final_hyphen_demerits_code: 236, 237, 238.
final_pass: 828, 854, 863, 873.
final_widow_penalty: 814, 815, 876,* 877,* 890.
find_font_dimen: 425, 578, 1042, 1253.
fingers: 511.
finite_shrink: 825, 826.
fire_up: 1005, 1012.
firm_up_the_line: 340, 362, 363, 538.
first: 30, 31,* 35, 36,* 37,* 71, 83, 87, 88, 328, 329, 331, 355, 360,* 362, 363, 374, 483, 531, 538.
first_child: 960, 963, 964.

first_count: 54, 315, 316, 317.
first_fit: 953, 957, 966.
first_indent: 847, 849, 889.
first_mark: 382, 383, 1012, 1016.
`\firstmark` primitive: 384.
first_mark_code: 382, 384, 385.
first_text_char: 19, 24.
first_width: 847, 849, 850, 889.
fit_class: 830, 836, 845, 846, 852, 853, 855, 859.
fitness: 819, 845, 859, 864.
fix_date_and_time: 241,* 1332,* 1337.
fix_language: 1034, 1376.
fix_word: 541, 542, 547, 548, 571.
float: 109,* 114, 186, 625, 634, 809.
float_constant: 109,* 186, 619, 625, 629, 1123, 1125.
float_cost: 140, 188, 1008, 1100.
floating_penalty: 140, 236, 1068, 1100.
`\floatingpenalty` primitive: 238.
floating_penalty_code: 236, 237, 238.
flush_char: 42, 180, 195, 692, 695.
flush_list: 123, 200, 324, 372, 396, 407, 801, 903, 960, 1279, 1297, 1370.
flush_math: 718, 776, 1195.
flush_node_list: 199, 202, 275, 639, 698, 718, 731, 732, 742, 800, 816,* 879,* 883, 903, 918, 968, 992, 999, 1078, 1105, 1120, 1121, 1375.
flush_string: 44, 264, 1260, 1279, 1328.
flushable_string: 1257, 1260.
fmem_ptr: 425, 549, 552, 566, 569, 570, 576, 578, 579, 580, 1320, 1321, 1323, 1334.
fmt_file: 524, 1305, 1306, 1308, 1327, 1328, 1329, 1337, 1379.*
fnt_def1: 585, 586, 602.
fnt_def2: 585.
fnt_def3: 585.
fnt_def4: 585.
fnt_num_0: 585, 586, 621.
fnt1: 585, 586, 621.
fnt2: 585.
fnt3: 585.
fnt4: 585.
font: 134, 143, 144, 174, 176, 193, 206, 267, 548, 582, 620, 654, 681, 709, 715, 724, 841, 842, 866, 867, 870, 871, 896, 897, 898, 903, 908, 911, 1034, 1038, 1113, 1147.
font metric files: 539.
font parameters: 700, 701.
`Font x has only...`: 579.
`Font x=xx not loadable...`: 561.
`Font x=xx not loaded...`: 567.
`\font` primitive: 265.
font_area: 549, 552, 576, 602, 603, 1260, 1322, 1323.
font_base: 11,* 12, 111, 134, 174, 176, 222, 232, 548, 551, 602, 621, 643, 1260, 1320, 1321, 1334.
font_bc: 549, 552, 576, 582, 708, 722, 1036, 1322, 1323.
font_bchar: 549, 552, 576, 897, 898, 915, 1032, 1034, 1322, 1323.
font_check: 549, 568, 602, 1322, 1323.
`\fontdimen` primitive: 265.
font_dsize: 472, 549, 552, 568, 602, 1260, 1261, 1322, 1323.
font_ec: 549, 552, 576, 582, 708, 722, 1036, 1322, 1323.
font_false_bchar: 549, 552, 576, 1032, 1034, 1322, 1323.
font_glue: 549, 552, 576, 578, 1042, 1322, 1323.
font_id_base: 222, 234, 256, 415, 548, 1257.
font_id_text: 234, 256, 267, 579, 1257, 1322.
font_in_short_display: 173, 174, 193, 663, 864, 1339.*
font_index: 548, 549, 560, 906, 1032, 1211.
font_info: 11,* 425, 548, 549, 550, 552, 554, 557, 558, 560, 566, 569, 571, 573, 574, 575, 578, 580, 700, 701, 713, 741, 752, 909, 1032, 1039, 1042, 1211, 1253, 1320, 1321, 1339.*
font_max: 11,* 111, 174, 176, 548, 551, 566, 1321, 1334.
font_mem_size: 11,* 548, 566, 580, 1321, 1334.
font_name: 472, 549, 552, 576, 581, 602, 603, 1260, 1261, 1322, 1323.
`\fontname` primitive: 468.
font_name_code: 468, 469, 471, 472.
font_params: 549, 552, 576, 578, 579, 580, 1195, 1322, 1323.
font_ptr: 549, 552, 566, 576, 578, 643, 1260, 1320, 1321, 1334.
font_size: 472, 549, 552, 568, 602, 1260, 1261, 1322, 1323.
font_used: 549, 551, 621, 643.
`FONTx`: 1257.
`for accent`: 191.
`Forbidden control sequence...`: 338.
force_eof: 331, 361, 362, 378.
format_area_length: 520, 524.
format_default_length: 520, 522, 523, 524, 1379.*
format_ext_length: 520, 523, 524, 1379.*
format_extension: 520, 529, 1328.
format_ident: 35, 61, 536, 1299, 1300, 1301, 1326, 1327, 1328, 1337, 1379.*
forward: 4,* 78, 218, 281, 340, 366, 409, 618, 692, 693, 720, 774, 800.

found: 15, 125, 128, 129, 259, 341, 354, 356, 389, 392, 394, 448, 455, 473, 475, 477, 524, 607, 609, 612, 613, 614, 645, 706, 708, 720, 895, 923, 931, 934, 941, 953, 955, 1138, 1146, 1147, 1148, 1236, 1237.
found1: 15, 895, 902, 1302, 1315.
found2: 15, 895, 903, 1302, 1316.
four_choices: 113.
four_quarters: 113, 548, 549, 554, 555, 560, 649, 683, 684, 706, 709, 712, 724, 738, 749, 906, 1032, 1123, 1302, 1303.
fraction_noad: 683, 687, 690, 698, 733, 761, 1178, 1181.
fraction_noad_size: 683, 698, 761, 1181.
fraction_rule: 704, 705, 735, 747.
free: 165, 167, 168, 169, 170, 171.
free_avail: 121, 202, 204, 217, 400, 452, 772, 915, 1036, 1226, 1288.
free_node: 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, 1186, 1187, 1201, 1335, 1358.
freeze_page_specs: 987, 1001, 1008.
frozen_control_sequence: 222, 258, 1215, 1314, 1318, 1319.
frozen_cr: 222, 339, 780, 1132.
frozen_dont_expand: 222, 258, 369.
frozen_end_group: 222, 265, 1065.
frozen_end_template: 222, 375, 780.
frozen_endv: 222, 375, 380, 780.
frozen_fi: 222, 336, 491.
frozen_null_font: 222, 553.
frozen_protection: 222, 1215, 1216.
frozen_relax: 222, 265, 379.
frozen_right: 222, 1065, 1188.
Fuchs, David Raymond: 2*, 583, 591.
`\futurelet` primitive: 1219.
g: 47, 182, 560, 592, 649, 668, 706, 716.
g_order: 619, 625, 629, 634.
g_sign: 619, 625, 629, 634.
garbage: 162, 467, 470, 960, 1183, 1192, 1279.
`\gdef` primitive: 1208.
geq_define: 279, 782, 1077, 1214.
geq_word_define: 279, 288, 1013, 1214.
get: 26, 29, 31*, 33*, 485, 538, 564, 1306.
get_avail: 120, 122, 204, 205, 216, 325, 337, 339, 369, 371, 372, 452, 473, 482, 582, 709, 772, 783, 784, 794, 908, 911, 938, 1064, 1065, 1226, 1371.
get_next: 76, 297, 332, 336, 340, 341, 357, 360*, 364, 365, 366, 369, 380, 381, 387, 389, 478, 494, 507, 644, 1038, 1126.
get_node: 125, 131, 136, 139, 144, 145, 147, 151, 152, 153, 156, 158, 206, 495, 607, 649, 668, 686, 688, 689, 716, 772, 798, 843, 844, 845, 864, 914, 1009, 1100, 1101, 1163, 1165, 1181, 1248, 1249, 1349, 1357.
get_preamble_token: 782, 783, 784.
get_r_token: 1215, 1218, 1221, 1224, 1225, 1257.
get_strings_started: 47, 51, 1332*
get_token: 76, 78, 88, 364, 365, 368, 369, 392, 399, 442, 452, 471, 473, 474, 476, 477, 479, 483, 782, 1027, 1138, 1215, 1221, 1252, 1268, 1271, 1294, 1371, 1372.
get_x_token: 364, 366, 372, 380, 381, 402, 404, 406, 407, 443, 444, 445, 452, 465, 479, 506, 526, 780, 935, 961, 1029, 1030, 1138, 1197, 1237, 1375.
get_x_token_or_active_char: 506.
give_err_help: 78, 89, 90, 1284.
global: 1214, 1218, 1241.
global definitions: 221, 279, 283.
`\global` primitive: 1208.
global_defs: 236, 782, 1214, 1218.
`\globaldefs` primitive: 238.
global_defs_code: 236, 237, 238.
glue_base: 220, 222, 224, 226, 227, 228, 229, 252, 782.
glue_node: 149, 152, 153, 175, 183, 202, 206, 424, 622, 631, 651, 669, 730, 732, 761, 816*, 817, 837, 856, 862*, 866, 879*, 881, 899, 903, 968, 972, 973, 988, 996, 997, 1000, 1106, 1107, 1108, 1147, 1202.
glue_offset: 135, 159, 186.
glue_ord: 150, 447, 619, 629, 646, 649, 668, 791.
glue_order: 135, 136, 159, 185, 186, 619, 629, 657, 658, 664, 672, 673, 676, 769, 796, 801, 807, 809, 810, 811, 1148.
glue_par: 224, 766.
glue_pars: 224.
glue_ptr: 149, 152, 153, 175, 189, 190, 202, 206, 424, 625, 634, 656, 671, 679, 732, 786, 793, 795, 802, 803, 809, 816*, 838, 868, 881, 969, 976, 996, 1001, 1004, 1148.
glue_ratio: 9*, 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_set: 135, 136, 159, 186, 625, 634, 657, 658, 664, 672, 673, 676, 807, 809, 810, 811, 1148.
glue_shrink: 159, 185, 796, 799, 801, 810, 811.
glue_sign: 135, 136, 159, 185, 186, 619, 629, 657, 658, 664, 672, 673, 676, 769, 796, 801, 807, 809, 810, 811, 1148.
glue_spec_size: 150, 151, 162, 164, 201, 716.

glue_stretch: 159, 185, 796, 799, 801, 810, 811.
glue_temp: 619, 625, 629, 634.
glue_val: 410, 411, 412, 413, 416, 417, 424, 427, 429, 430, 451, 461, 465, 782, 1060, 1228, 1236, 1237, 1238, 1240.
GNU Pascal: 9,* 10,* 25,* 27,* 28,* 34,* 36,* 96,* 109,* 241,* 1332,* 1380.*
`goal height`: 986, 987.
goto: 35, 81.
gpc_block_write: 597.*
gpc_byte: 25.*
gpc_close: 28.*
gpc_day: 241.*
gpc_execute: 4,* 1332,* 1380.*
gpc_get_time_stamp: 241.*
gpc_gpc: 4.*
gpc_halt: 1332.*
gpc_hour: 241.*
gpc_import: 4.*
gpc_install_signal_handler: 4,* 1381,* 1382.*
gpc_integer: 4,* 1381.*
gpc_io_result: 27.*
gpc_length: 36.*
gpc_minute: 241.*
gpc_month: 241.*
gpc_null: 1381,* 1382.*
gpc_only: 4.*
gpc_param_count: 36.*
gpc_param_str: 36,* 1380.*
gpc_short_real: 109.*
gpc_sig_int: 4,* 1381,* 1382.*
gpc_string: 36,* 1380.*
gpc_time_stamp: 241.*
gpc_trim: 27,* 1380.*
gpc_untyped_file: 27,* 28,* 532.*
gpc_write_str: 1380.*
gpc_year: 241.*
gr: 110, 113, 114, 135.
group_code: 269, 271, 274, 645, 1136.
gubed: 7.*
Guibas, Leonidas Ioannis: 2.*
g1: 1198, 1203.
g2: 1198, 1203, 1205.
h: 204, 259, 649, 668, 738, 929, 934, 944, 948, 953, 966, 970, 977, 994, 1086, 1091, 1123.
h_offset: 247, 617, 641.
`\hoffset` primitive: 248.
h_offset_code: 247, 248.
ha: 892, 896, 900, 903, 912.
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.
halfword: 108, 110, 113, 115, 130, 264, 277, 279, 280, 281, 297, 298, 300, 333, 341, 366, 389, 413, 464, 473, 549, 560, 577, 681, 791, 800, 821, 829, 830, 833, 847, 872, 877,* 892, 901, 906, 907, 1032, 1079, 1211, 1243, 1266, 1288.
halign: 208, 265, 266, 1094, 1130.
`\halign` primitive: 265.
handle_right_brace: 1067, 1068.
hang_after: 236, 240, 847, 849, 1070, 1149.
`\hangafter` primitive: 238.
hang_after_code: 236, 237, 238, 1070.
hang_indent: 247, 847, 848, 849, 1070, 1149.
`\hangindent` primitive: 248.
hang_indent_code: 247, 248, 1070.
hanging indentation: 847.
hash: 234, 256, 257, 259, 260, 1318, 1319.
hash_base: 220, 222, 256, 257, 259, 262, 263, 1257, 1314, 1318, 1319.
hash_brace: 473, 476.
hash_is_full: 256, 260.
hash_prime: 12, 14, 259, 261, 1307, 1308.
hash_size: 12, 14, 222, 260, 261, 1334.
hash_used: 256, 258, 260, 1318, 1319.
hb: 892, 897, 898, 900, 903.
hbadness: 236, 660, 666, 667.
`\hbadness` primitive: 238.
hbadness_code: 236, 237, 238.
`\hbox` primitive: 1071.
hbox_group: 269, 274, 1083, 1085.
hc: 892, 893, 897, 898, 900, 901, 919, 920, 923, 930, 931, 934, 937, 939, 960, 962, 963, 965.
hchar: 905, 906, 908, 909.
hd: 649, 654, 706, 708, 709, 712.
head: 212, 213, 215, 216, 217, 424, 718, 776, 796, 799, 805, 812, 814, 816,* 1026, 1054, 1080, 1081, 1086, 1091, 1096, 1100, 1105, 1113, 1119, 1121, 1145, 1159, 1168, 1176, 1181, 1184, 1185, 1187, 1191.
head_field: 212, 213, 218.
head_for_vmode: 1094, 1095.
header: 542.
Heckenbach, Frank: 31.*
Hedrick, Charles Locke: 3.
height: 135, 136, 138, 139, 140, 184, 187, 188, 463, 554, 622, 624, 626, 629, 631, 632, 635, 637, 640, 641, 649, 653, 656, 670, 672, 679, 704, 706, 709, 711, 713, 727, 730, 735, 736, 737, 738, 739, 742, 745, 746, 747, 749, 750, 751, 756, 757, 759, 768, 769, 796, 801, 804, 806, 807, 809, 810, 811, 969, 973, 981, 986, 1001, 1002, 1008, 1009, 1010, 1021, 1087, 1100.

height: 463.
height_base: 550, 552, 554, 566, 571, 1322, 1323.
height_depth: 554, 654, 708, 709, 712, 1125.
height_index: 543, 554.
height_offset: 135, 416, 417, 769, 1247.
height_plus_depth: 712, 714.
held over for next output: 986.
help_line: 79;* 89, 90, 336, 1106.
help_ptr: 79;* 80;* 89, 90.
help0: 79;* 1252, 1293.
help1: 79;* 93, 95, 288, 408, 428, 454, 476, 486, 500, 503, 510, 960, 961, 962, 963, 1066, 1080, 1099, 1121, 1132, 1135, 1159, 1177, 1192, 1212, 1213, 1232, 1237, 1243, 1244, 1258, 1283, 1304.
help2: 72, 79;* 88, 89, 94, 95, 288, 346, 373, 433, 434, 435, 436, 437, 442, 445, 460, 475, 476, 577, 579, 641, 936, 937, 978, 1015, 1027, 1047, 1068, 1080, 1082, 1095, 1106, 1120, 1129, 1166, 1197, 1207, 1225, 1236, 1241, 1259, 1372.
help3: 72, 79;* 98, 336, 396, 415, 446, 479, 776, 783, 784, 792, 993, 1009, 1024, 1028, 1078, 1084, 1110, 1127, 1183, 1195, 1293.
help4: 79;* 89, 338, 398, 403, 418, 456, 567, 723, 976, 1004, 1050, 1283.
help5: 79;* 370, 561, 826, 1064, 1069, 1128, 1215, 1293.
help6: 79;* 395, 459, 1128, 1161.
Here is how much...: 1334.
hex_to_cur_chr: 352, 355.
hex_token: 438, 444.
hf: 892, 896, 897, 898, 903, 908, 909, 910, 911, 915, 916.
\hfil primitive: 1058.
\hfilneg primitive: 1058.
\hfill primitive: 1058.
hfuzz: 247, 666.
\hfuzz primitive: 248.
hfuzz_code: 247, 248.
hh: 110, 113, 114, 118, 133, 182, 213, 219, 221, 268, 686, 742, 1163, 1165, 1181, 1186, 1305, 1306.
hi: 112;* 232, 1232.
hi_mem_min: 116, 118, 120, 125, 126, 134, 164, 165, 167, 168, 171, 172, 176, 293, 639, 1311, 1312, 1334.
hi_mem_stat_min: 162, 164, 1312.
hi_mem_stat_usage: 162, 164.
history: 76, 77, 82, 93, 95, 245, 1332;* 1335.
hlist_node: 135, 136, 137, 138, 148, 159, 175, 183, 184, 202, 206, 505, 618, 619, 622, 631, 644, 649, 651, 669, 681, 807, 810, 814, 841, 842, 866, 870, 871, 968, 973, 993, 1000, 1074, 1080, 1087, 1110, 1147, 1203.
hlist_out: 592, 615, 616, 618, 619, 620, 623, 628, 629, 632, 637, 638, 640, 693, 1373.
hlp1: 79.*
hlp2: 79.*
hlp3: 79.*
hlp4: 79.*
hlp5: 79.*
hlp6: 79.*
hmode: 211, 218, 416, 501, 786, 787, 796, 799, 1030, 1045, 1046, 1048, 1056, 1057, 1071, 1073, 1076, 1079, 1083, 1086, 1091, 1092, 1093, 1094, 1096, 1097, 1109, 1110, 1112, 1116, 1117, 1119, 1122, 1130, 1137, 1200, 1243, 1377.
hmove: 208, 1048, 1071, 1072, 1073.
hn: 892, 897, 898, 899, 902, 912, 913, 915, 916, 917, 919, 923, 930, 931.
ho: 112;* 235, 414, 1151, 1154.
hold_head: 162, 306, 779, 783, 784, 794, 808, 905, 906, 913, 914, 915, 916, 917, 1014, 1017.
holding_inserts: 236, 1014.
\holdinginserts primitive: 238.
holding_inserts_code: 236, 237, 238.
hpack: 162, 236, 644, 645, 646, 647, 649, 661, 709, 715, 720, 727, 737, 748, 754, 756, 796, 799, 804, 806, 889, 1062, 1086, 1125, 1194, 1199, 1201, 1204.
hrule: 208, 265, 266, 463, 1046, 1056, 1084, 1094, 1095.
\hrule primitive: 265.
hsize: 247, 847, 848, 849, 1054, 1149.
\hsize primitive: 248.
hsize_code: 247, 248.
hskip: 208, 1057, 1058, 1059, 1078, 1090.
\hskip primitive: 1058.
\hss primitive: 1058.
\ht primitive: 416.
hu: 892, 893, 897, 898, 901, 903, 905, 907, 908, 910, 911, 912, 915, 916.
Huge page...: 641.
hyf: 900, 902, 905, 908, 909, 913, 914, 919, 920, 923, 924, 932, 960, 961, 962, 963, 965.
hyf_bchar: 892, 897, 898, 903.
hyf_char: 892, 896, 913, 915.
hyf_distance: 920, 921, 922, 924, 943, 944, 945, 1324, 1325.
hyf_next: 920, 921, 924, 943, 944, 945, 1324, 1325.
hyf_node: 912, 915.
hyf_num: 920, 921, 924, 943, 944, 945, 1324, 1325.
hyph_count: 926, 928, 940, 1324, 1325, 1334.
hyph_data: 209, 1210, 1250, 1251, 1252.
hyph_list: 926, 928, 929, 932, 933, 934, 940, 941, 1324, 1325.

hyph_pointer: 925, 926, 927, 929, 934.
hyph_size: 12, 925, 928, 930, 933, 939, 940, 1307, 1308, 1324, 1325, 1334.
hyph_word: 926, 928, 929, 931, 934, 940, 941, 1324, 1325.
hyphen_char: 426, 549, 552, 576, 891, 896, 1035, 1117, 1253, 1322, 1323.
`\hyphenchar` primitive: 1254.
hyphen_passed: 905, 906, 909, 913, 914.
hyphen_penalty: 145, 236, 869.
`\hyphenpenalty` primitive: 238.
hyphen_penalty_code: 236, 237, 238.
hyphenate: 894, 895.
hyphenated: 819, 820, 829, 846, 859, 869, 873.
`Hyphenation trie...`: 1324.
`\hyphenation` primitive: 1250.
i: 19, 315, 587, 649, 738, 749, 901, 1123, 1348, 1380*.
`I can't find file x`: 530.
`I can't find PLAIN...`: 524.
`I can't go on...`: 95.
`I can't read TEX.POOL`: 51.
`I can't write on file x`: 530.
id_byte: 587, 617, 642*.
id_lookup: 259, 264, 356, 374.
ident_val: 410, 415, 465, 466.
`\ifcase` primitive: 487.
if_case_code: 487, 488, 501.
if_cat_code: 487, 488, 501.
`\ifcat` primitive: 487.
`\if` primitive: 487.
if_char_code: 487, 501, 506.
if_code: 489, 495, 510.
`\ifdim` primitive: 487.
if_dim_code: 487, 488, 501.
`\ifeof` primitive: 487.
if_eof_code: 487, 488, 501.
`\iffalse` primitive: 487.
if_false_code: 487, 488, 501.
`\ifhbox` primitive: 487.
if_hbox_code: 487, 488, 501, 505.
`\ifhmode` primitive: 487.
if_hmode_code: 487, 488, 501.
`\ifinner` primitive: 487.
if_inner_code: 487, 488, 501.
`\ifnum` primitive: 487.
if_int_code: 487, 488, 501, 503.
if_limit: 489, 490, 495, 496, 497, 498, 510.
if_line: 489, 490, 495, 496, 1335.
if_line_field: 489, 495, 496, 1335.
`\ifmmode` primitive: 487.
if_mmode_code: 487, 488, 501.
if_node_size: 489, 495, 496, 1335.
`\ifodd` primitive: 487.
if_odd_code: 487, 488, 501.
if_test: 210, 336, 366, 367, 487, 488, 494, 498, 503, 1335.
`\iftrue` primitive: 487.
if_true_code: 487, 488, 501.
`\ifvbox` primitive: 487.
if_vbox_code: 487, 488, 501.
`\ifvmode` primitive: 487.
if_vmode_code: 487, 488, 501.
`\ifvoid` primitive: 487.
if_void_code: 487, 488, 501, 505.
`\ifx` primitive: 487.
ifx_code: 487, 488, 501.
ignore: 207, 232, 332, 345.
ignore_depth: 212, 215, 219, 679, 787, 1025, 1056, 1083, 1099, 1167.
ignore_spaces: 208, 265, 266, 1045.
`\ignorespaces` primitive: 265.
`Illegal magnification...`: 288, 1258.
`Illegal math \disc...`: 1120.
`Illegal parameter number...`: 479.
`Illegal unit of measure`: 454, 456, 459.
`\immediate` primitive: 1344.
immediate_code: 1344, 1346, 1348.
`IMPOSSIBLE`: 262.
`Improper \halign...`: 776.
`Improper \hyphenation...`: 936.
`Improper \prevdepth`: 418.
`Improper \setbox`: 1241.
`Improper \spacefactor`: 418.
`Improper 'at' size...`: 1259.
`Improper alphabetic constant`: 442.
`Improper discretionary list`: 1121.
`in`: 458.
in_open: 304, 328, 329, 331.
in_state_record: 300, 301.
in_stream: 208, 1272, 1273, 1274.
`Incompatible glue units`: 408.
`Incompatible list...`: 1110.
`Incompatible magnification`: 288.
incompleat_noad: 212, 213, 718, 776, 1136, 1178, 1181, 1182, 1184, 1185.
`Incomplete \if...`: 336.
incr: 16, 31*, 36*, 42, 43, 45, 46, 53, 58, 59, 60, 65, 67, 70, 71, 82, 90, 98, 120, 122, 152, 153, 170, 182, 203, 216, 260, 274, 276, 280, 294, 311, 312, 321, 325, 328, 343, 347, 352, 354, 355, 356, 357, 360*, 362, 374, 392, 395, 397, 399, 400, 403, 407, 442, 452, 454, 464, 475, 476, 477, 494, 517, 519, 524, 531, 537*, 580, 598, 619, 629, 640, 642*, 645,

714, 798, 845, 877,* 897, 898, 910, 911, 914, 915, 923, 930, 931, 937, 939, 940, 941, 944, 954, 956, 962, 963, 964, 986, 1022, 1025, 1035, 1039, 1069, 1099, 1117, 1119, 1121, 1127, 1142, 1153, 1172, 1174, 1315, 1316, 1318, 1337, 1380.*
\indent primitive: 1088.
indent_in_hmode: 1092, 1093.
indented: 1091.
index: 300, 302, 303, 304, 307, 328, 329, 331.
index_field: 300, 302, 1131.
inf: 447, 448, 453.
inf_bad: 108, 157, 851, 852, 853, 856, 863, 974, 1005, 1017.
inf_penalty: 157, 761, 767, 816,* 829, 831, 974, 1005, 1013, 1203, 1205.
Infinite glue shrinkage...: 826, 976, 1004, 1009.
infinity: 445.
info: 118, 124, 126, 140, 164, 172, 200, 233, 275, 291, 293, 325, 337, 339, 357, 358, 369, 371, 374, 389, 391, 392, 393, 394, 397, 400, 423, 452, 466, 508, 605, 608, 609, 610, 611, 612, 613, 614, 615, 681, 689, 692, 693, 698, 720, 734, 735, 736, 737, 738, 742, 749, 754, 768, 769, 772, 779, 783, 784, 790, 793, 794, 797, 798, 801, 803, 821, 847, 848, 925, 932, 938, 981, 1065, 1076, 1093, 1149, 1151, 1168, 1181, 1185, 1186, 1191, 1226, 1248, 1249, 1289, 1312, 1339,* 1341, 1371.
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_l_hyf: 816,* 891, 892.
init_lft: 900, 903, 905, 908.
init_lig: 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_r_hyf: 816,* 891, 892.
init_row: 773, 785, 786.
init_span: 773, 786, 787, 791.
init_str_ptr: 39, 43, 517, 1310, 1332,* 1334.
init_terminal: 37,* 331.
init_trie: 891, 966, 1324.
INITEX: 8, 11,* 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,* 611, 620, 651, 654, 655, 832, 835, 851, 852, 867, 1030, 1034, 1035, 1036, 1039, 1041.
inner_noad: 682, 683, 690, 696, 698, 733, 761, 764, 1156, 1157, 1191.
input: 31,* 210, 366, 367, 376, 377.
\input primitive: 376.
input_command_ln: 36,* 37.*
input_file: 304.
\inputlineno primitive: 416.
input_line_no_code: 416, 417, 424.
input_ln: 30, 31,* 36,* 37,* 58, 71, 362, 485, 486, 538.
input_ptr: 301, 311, 312, 321, 322, 330, 331, 360,* 534, 1131, 1335.
input_stack: 84,* 301, 311, 321, 322, 534, 1131.
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.
ins_ptr: 140, 188, 202, 206, 1010, 1020, 1021, 1100.
ins_the_toks: 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_relax: 378, 379, 510.
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: 110, 113, 114, 140, 141, 157, 186, 213, 219, 236, 240, 242, 274, 278, 279, 413, 414, 489, 605, 725, 769, 772, 819, 1238, 1240, 1305, 1306, 1308, 1316.
int_base: 220, 230, 232, 236, 238, 239, 240, 242, 252, 253, 254, 268, 283, 288, 1013, 1070, 1139, 1145, 1315.
int_error: 91, 288, 433, 434, 435, 436, 437, 1243, 1244, 1258.
int_par: 236.
int_pars: 236.
int_val: 410, 411, 412, 413, 414, 416, 417, 418, 419, 422, 423, 424, 426, 427, 428, 429, 439, 440, 449, 461, 465, 1236, 1237, 1238, 1240.
integer: 3, 13, 19, 36,* 45, 47, 54, 59, 60, 63, 65, 66, 67, 69, 79,* 82, 91, 94, 96,* 100, 101, 102, 105,

106, 107, 108, 109,* 110, 113, 117, 125, 158, 163, 172, 173, 174, 176, 177, 178, 181, 182, 211, 212, 218, 225, 237, 247, 256, 259, 262, 278, 279, 286, 292, 304, 308, 309, 311, 315, 366, 410, 440, 448, 450, 482, 489, 493, 494, 498, 518, 519, 523, 549, 550, 560, 578, 592, 595, 600, 601, 607, 615, 616, 619, 629, 638, 645, 646, 661, 691, 694, 699, 706, 716, 717, 726, 738, 752, 764, 815, 828, 829, 830, 833, 872, 877,* 892, 912, 922, 966, 970, 980, 982, 994, 1012, 1030, 1032, 1068, 1075, 1079, 1084, 1091, 1117, 1119, 1138, 1151, 1155, 1194, 1211, 1302, 1303, 1331, 1333,* 1338,* 1348, 1370, 1380.*

inter_line_penalty: 236, 890.

`\interlinepenalty` primitive: 238.

inter_line_penalty_code: 236, 237, 238.

interaction: 71, 72, 73, 74, 75, 82, 84,* 86, 90, 92, 93, 98, 360,* 363, 484, 530, 1265, 1283, 1293, 1294, 1297, 1326, 1327, 1328, 1335.

internal_font_number: 548, 549, 550, 560, 577, 578, 581, 582, 602, 616, 649, 706, 709, 711, 712, 715, 724, 738, 830, 862,* 892, 1032, 1113, 1123, 1138, 1211, 1257.

interrupt: 96,* 97, 98, 1031, 1381,* 1382.*

`Interruption`: 98.

`interwoven alignment preambles...`: 324, 782, 789, 791, 1131.

`Invalid code`: 1232.

invalid_char: 207, 232, 344.

invalid_code: 22, 24, 232.

is_char_node: 134, 174, 183, 202, 205, 424, 620, 630, 651, 669, 715, 720, 721, 756, 805, 816,* 837, 841, 842, 866, 867, 868, 870, 871, 879,* 896, 897, 899, 903, 1036, 1040, 1080, 1081, 1105, 1113, 1121, 1147, 1202.

is_empty: 124, 127, 169, 170.

is_hex: 352, 355.

is_running: 138, 176, 624, 633, 806.

issue_message: 1276, 1279.

ital_corr: 208, 265, 266, 1111, 1112.

italic correction: 543.

italic_base: 550, 552, 554, 566, 571, 1322, 1323.

italic_index: 543.

its_all_over: 1045, 1054, 1335.

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.

Japanese characters: 134, 585.

Jensen, Kathleen: 10.*

Jerabek, Emil: 597.*

`job aborted`: 360.*

`job aborted, file error...`: 530.

job_name: 92, 471, 472, 527, 528, 529, 532,* 534, 537,* 1257, 1328, 1335.

`\jobname` primitive: 468.

job_name_code: 468, 470, 471, 472.

jump_out: 81, 82, 84,* 93.

just_box: 814, 888, 889, 1146, 1148.

just_open: 480, 483, 1275.

k: 45, 46, 47, 64, 65, 67, 69, 71, 102, 163, 259, 264, 341, 363, 407, 450, 464, 519, 523, 525, 530, 534, 560, 587, 602, 607, 638, 705, 906, 929, 934, 960, 966, 1079, 1211, 1302, 1303, 1333,* 1338,* 1348, 1368.

kern: 208, 545, 1057, 1058, 1059.

`\kern` primitive: 1058.

kern_base: 550, 552, 557, 566, 573, 576, 1322, 1323.

kern_base_offset: 557, 566, 573.

kern_break: 866.

kern_flag: 545, 741, 753, 909, 1040.

kern_node: 155, 156, 183, 202, 206, 424, 622, 631, 651, 669, 721, 730, 732, 761, 837, 841, 842, 856, 866, 868, 870, 871, 879,* 881, 896, 897, 899, 968, 972, 973, 976, 996, 997, 1000, 1004, 1106, 1107, 1108, 1121, 1147.

kk: 450, 452.

Knuth, Donald Ervin: 2,* 86, 693, 813, 891, 925, 997, 1154, 1371.

Kuebart, Joachim: 37.*

l: 47, 259, 264, 276, 281, 292, 315, 494, 497, 534, 601, 615, 668, 830, 901, 944, 953, 960, 1138, 1194, 1236, 1302, 1338,* 1376.

l_hyf: 891, 892, 894, 899, 902, 923, 1362.

label: 4.*

language: 236, 934, 1034, 1376.

`\language` primitive: 238.

language_code: 236, 237, 238.

language_node: 1341, 1356, 1357, 1358, 1362, 1373, 1376, 1377.

large_attempt: 706.

large_char: 683, 691, 697, 706, 1160.

large_fam: 683, 691, 697, 706, 1160.

last: 30, 31,* 35, 36,* 37,* 71, 83, 87, 88, 331, 360,* 363, 483, 524, 531.

last_active: 819, 820, 832, 835, 844, 854, 860, 861, 863, 864, 865, 873, 874, 875.

last_badness: 424, 646, 648, 649, 660, 664, 667, 668, 674, 676, 678.

last_bop: 592, 593, 640, 642.*

`\lastbox` primitive: 1071.

last_box_code: 1071, 1072, 1079.

last_glue: 424, 982, 991, 996, 1017, 1106, 1335.

last_ins_ptr: 981, 1005, 1008, 1018, 1020.

last_item: 208, 413, 416, 417, 1048.

last_kern: 424, 982, 991, 996.
`\lastkern` primitive: 416.
last_penalty: 424, 982, 991, 996.
`\lastpenalty` primitive: 416.
`\lastskip` primitive: 416.
last_special_line: 847, 848, 849, 850, 889.
last_text_char: 19, 24.
lc_code: 230, 232, 891, 896, 897, 898, 937, 962.
`\lccode` primitive: 1230.
lc_code_base: 230, 235, 1230, 1231, 1286, 1287, 1288.
leader_box: 619, 626, 628, 629, 635, 637.
leader_flag: 1071, 1073, 1078, 1084.
leader_ht: 629, 635, 636, 637.
leader_ptr: 149, 152, 153, 190, 202, 206, 626, 635, 656, 671, 816,* 1078.
leader_ship: 208, 1071, 1072, 1073.
leader_wd: 619, 626, 627, 628.
leaders: 1374.
`Leaders not followed by...`: 1078.
`\leaders` primitive: 1071.
least_cost: 970, 974, 980.
least_page_cost: 980, 987, 1005, 1006.
`\left` primitive: 1188.
left_brace: 207, 289, 294, 298, 347, 357, 403, 473, 476, 777, 1063, 1150, 1226.
left_brace_limit: 289, 325, 392, 394, 399.
left_brace_token: 289, 403, 1127, 1226, 1371.
left_delimiter: 683, 696, 697, 737, 748, 1163, 1181, 1182.
left_edge: 619, 627, 629, 632, 637.
left_hyphen_min: 236, 1091, 1200, 1376, 1377.
`\lefthyphenmin` primitive: 238.
left_hyphen_min_code: 236, 237, 238.
left_noad: 687, 690, 696, 698, 725, 728, 733, 760, 761, 762, 1185, 1188, 1189, 1191.
left_right: 208, 1046, 1188, 1189, 1190.
left_skip: 224, 827, 880, 887.
`\leftskip` primitive: 226.
left_skip_code: 224, 225, 226, 887.
length: 40, 46, 259, 537,* 602, 931, 941, 1280.
length of lines: 847.
`\leqno` primitive: 1141.
let: 209, 1210, 1219, 1220, 1221.
`\let` primitive: 1219.
letter: 207, 232, 262, 289, 291, 294, 298, 347, 354, 356, 935, 961, 1029, 1030, 1038, 1090, 1124, 1151, 1154, 1160.
letter_token: 289, 445.
level: 410, 413, 415, 418, 428, 461.
level_boundary: 268, 270, 274, 282.
level_one: 221, 228, 232, 254, 264, 272, 277, 278, 279, 280, 281, 283, 780, 1304, 1335, 1369.
level_zero: 221, 222, 272, 276, 280.
lf: 540, 560, 565, 566, 575, 576.
lft_hit: 906, 907, 908, 910, 911, 1033, 1035, 1040.
lh: 110, 113, 114, 118, 213, 219, 256, 540, 541, 560, 565, 566, 568, 685, 950.
Liang, Franklin Mark: 2,* 919.
lig_char: 143, 144, 193, 206, 652, 841, 842, 866, 870, 871, 898, 903, 1113.
lig_kern: 544, 545, 549.
lig_kern_base: 550, 552, 557, 566, 571, 573, 576, 1322, 1323.
lig_kern_command: 541, 545.
lig_kern_restart: 557, 741, 752, 909, 1039.
lig_kern_restart_end: 557.
lig_kern_start: 557, 741, 752, 909, 1039.
lig_ptr: 143, 144, 175, 193, 202, 206, 896, 898, 903, 907, 910, 911, 1037, 1040.
lig_stack: 907, 908, 910, 911, 1032, 1034, 1035, 1036, 1037, 1038, 1040.
lig_tag: 544, 569, 741, 752, 909, 1039.
lig_trick: 162, 652.
ligature_node: 143, 144, 148, 175, 183, 202, 206, 622, 651, 752, 841, 842, 866, 870, 871, 896, 897, 899, 903, 1113, 1121, 1147.
ligature_present: 906, 907, 908, 910, 911, 1033, 1035, 1037, 1040.
limit: 300, 302, 303, 307, 318, 328, 330, 331, 343, 348, 350, 351, 352, 354, 355, 356, 360,* 362, 363, 483, 537,* 538, 1337.
`Limit controls must follow...`: 1159.
limit_field: 35, 87, 300, 302, 534.
limit_switch: 208, 1046, 1156, 1157, 1158.
limits: 682, 696, 733, 749, 1156, 1157.
`\limits` primitive: 1156.
line: 84,* 216, 304, 313, 328, 329, 331, 362, 424, 494, 495, 538, 663, 675, 1025, 1380.*
line_break: 162, 814, 815, 828, 839, 848, 862,* 863, 866, 876,* 894, 934, 967, 970, 982, 1096, 1145.
line_diff: 872, 875.
line_number: 819, 820, 833, 835, 845, 846, 850, 864, 872, 874, 875.
line_penalty: 236, 859.
`\linepenalty` primitive: 238.
line_penalty_code: 236, 237, 238.
line_skip: 224, 247.
`\lineskip` primitive: 226.
line_skip_code: 149, 152, 224, 225, 226, 679.
line_skip_limit: 247, 679.
`\lineskiplimit` primitive: 248.
line_skip_limit_code: 247, 248.

line_stack: 304, 328, 329.
line_width: 830, 850, 851.
link: 118, 120, 121, 122, 123, 124, 125, 126, 130, 133, 134, 135, 140, 143, 150, 164, 168, 172, 174, 175, 176, 182, 202, 204, 212, 214, 218, 223, 233, 292, 295, 306, 319, 323, 339, 357, 358, 366, 369, 371, 374, 389, 390, 391, 394, 396, 397, 400, 407, 452, 464, 466, 467, 470, 478, 489, 495, 496, 497, 508, 605, 607, 609, 611, 615, 620, 622, 630, 649, 651, 652, 654, 655, 666, 669, 679, 681, 689, 705, 711, 715, 718, 719, 720, 721, 727, 731, 732, 735, 737, 738, 739, 747, 748, 751, 752, 753, 754, 755, 756, 759, 760, 761, 766, 767, 770, 772, 778, 779, 783, 784, 786, 790, 791, 793, 794, 795, 796, 797, 798, 799, 801, 802, 803, 804, 805, 806, 807, 808, 809, 812, 814, 816* 819, 821, 822, 829, 830, 837, 840, 843, 844, 845, 854, 857, 858, 860, 861, 862* 863, 864, 865, 866, 867, 869, 873, 874, 875, 877* 879* 880, 881, 882, 883, 884, 885, 886, 887, 888, 890, 894, 896, 897, 898, 899, 903, 905, 906, 907, 908, 910, 911, 913, 914, 915, 916, 917, 918, 932, 938, 960, 968, 969, 970, 973, 979, 980, 981, 986, 988, 991, 994, 998, 999, 1000, 1001, 1005, 1008, 1009, 1014, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1026, 1035, 1036, 1037, 1040, 1041, 1043, 1064, 1065, 1076, 1081, 1086, 1091, 1100, 1101, 1105, 1110, 1119, 1120, 1121, 1123, 1125, 1146, 1155, 1168, 1181, 1184, 1185, 1186, 1187, 1191, 1194, 1196, 1199, 1204, 1205, 1206, 1226, 1279, 1288, 1297, 1311, 1312, 1335, 1339* 1341, 1349, 1368, 1371, 1375.
list_offset: 135, 649, 769, 1018.
list_ptr: 135, 136, 184, 202, 206, 619, 623, 629, 632, 658, 663, 664, 668, 673, 676, 709, 711, 715, 721, 739, 747, 751, 807, 977, 979, 1021, 1087, 1100, 1110, 1146, 1199.
list_state_record: 212, 213.
list_tag: 544, 569, 570, 708, 740, 749.
ll: 953, 956.
llink: 124, 126, 127, 129, 130, 131, 145, 149, 164, 169, 772, 819, 821, 1312.
lo_mem_max: 116, 120, 125, 126, 164, 165, 167, 169, 170, 171, 172, 178, 639, 1311, 1312, 1323, 1334.
lo_mem_stat_max: 162, 164, 1312.
load_fmt_file: 1303, 1337, 1379*.
loc: 36* 37* 87, 300, 302, 303, 307, 312, 314, 318, 319, 323, 325, 328, 330, 331, 343, 348, 350, 351, 352, 354, 356, 357, 358, 360* 362, 369, 390, 483, 524, 537* 538, 1026, 1027, 1337.
loc_field: 35, 36* 300, 302, 1131.
local_base: 220, 224, 228, 230, 252.
location: 605, 607, 612, 613, 614, 615.
log_file: 54, 56, 75, 534, 1333*.
log_name: 532* 534, 1333*.
log_only: 54, 57, 58, 62, 75, 98, 360* 534, 1328, 1370.
log_opened: 92, 93, 527, 528, 534, 535, 1265, 1333* 1334.
`\long` primitive: 1208.
long_call: 210, 275, 366, 387, 389, 392, 399, 1295.
long_help_seen: 1281, 1282, 1283.
long_outer_call: 210, 275, 366, 387, 389, 1295.
long_state: 339, 387, 391, 392, 395, 396, 399.
loop: 15, 16.
`Loose \hbox...`: 660.
`Loose \vbox...`: 674.
loose_fit: 817, 834, 852.
looseness: 236, 848, 873, 875, 1070.
`\looseness` primitive: 238.
looseness_code: 236, 237, 238, 1070.
`\lower` primitive: 1071.
`\lowercase` primitive: 1286.
lq: 592, 627, 636.
lr: 592, 627, 636.
lx: 619, 626, 627, 628, 629, 635, 636, 637.
m: 47, 65, 158, 211, 218, 292, 315, 389, 413, 440, 482, 498, 577, 649, 668, 706, 716, 717, 1079, 1105, 1194, 1338*
mac_param: 207, 291, 294, 298, 347, 474, 477, 479, 783, 784, 1045.
macro: 307, 314, 319, 323, 324, 390.
macro_call: 291, 366, 380, 382, 387, 388, 389, 391.
macro_def: 473, 477.
mag: 236, 240, 288, 457, 585, 587, 588, 590, 617, 642*
`\mag` primitive: 238.
mag_code: 236, 237, 238, 288.
mag_set: 286, 287, 288.
magic_offset: 764, 765, 766.
main_control: 1029, 1030, 1032, 1040, 1041, 1052, 1054, 1055, 1056, 1057, 1126, 1134, 1208, 1290, 1332* 1337, 1344, 1347.
main_f: 1032, 1034, 1035, 1036, 1037, 1038, 1039, 1040.
main_i: 1032, 1036, 1037, 1039, 1040.
main_j: 1032, 1039, 1040.
main_k: 1032, 1034, 1039, 1040, 1042.
main_lig_loop: 1030, 1034, 1037, 1038, 1039, 1040.
main_loop: 1030.
main_loop_lookahead: 1030, 1034, 1036, 1037, 1038.
main_loop_move: 1030, 1034, 1036, 1040.
main_loop_move_lig: 1030, 1034, 1036, 1037.

main_loop_wrapup: 1030, 1034, 1039, 1040.
main_p: 1032, 1035, 1037, 1040, 1041, 1042, 1043, 1044.
main_s: 1032, 1034.
major_tail: 912, 914, 917, 918.
make_accent: 1122, 1123.
make_box: 208, 1071, 1072, 1073, 1079, 1084.
make_fraction: 733, 734, 743.
make_left_right: 761, 762.
make_mark: 1097, 1101.
make_math_accent: 733, 738.
make_name_string: 525, 532*
make_op: 733, 749.
make_ord: 733, 752.
make_over: 733, 734.
make_radical: 733, 734, 737.
make_scripts: 754, 756.
make_string: 43, 48, 52, 260, 517, 525, 939, 1257, 1279, 1328, 1333*
make_under: 733, 735.
make_vcenter: 733, 736.
mark: 208, 265, 266, 1097.
\mark primitive: 265.
mark_node: 141, 148, 175, 183, 202, 206, 647, 651, 730, 761, 866, 899, 968, 973, 979, 1000, 1014, 1101.
mark_ptr: 141, 142, 196, 202, 206, 979, 1016, 1101.
mark_text: 307, 314, 323, 386.
mastication: 341.
match: 207, 289, 291, 292, 294, 391, 392.
match_chr: 292, 294, 389, 391, 400.
match_token: 289, 391, 392, 393, 394, 476.
matching: 305, 306, 339, 391.
Math formula deleted...: 1195.
math_ac: 1164, 1165.
math_accent: 208, 265, 266, 1046, 1164.
\mathaccent primitive: 265.
\mathbin primitive: 1156.
math_char: 681, 692, 720, 722, 724, 738, 741, 749, 752, 753, 754, 1151, 1155, 1165.
\mathchar primitive: 265.
\mathchardef primitive: 1222.
math_char_def_code: 1222, 1223, 1224.
math_char_num: 208, 265, 266, 1046, 1151, 1154.
math_choice: 208, 265, 266, 1046, 1171.
\mathchoice primitive: 265.
math_choice_group: 269, 1172, 1173, 1174.
\mathclose primitive: 1156.
math_code: 230, 232, 236, 414, 1151, 1154.
\mathcode primitive: 1230.
math_code_base: 230, 235, 414, 1230, 1231, 1232, 1233.
math_comp: 208, 1046, 1156, 1157, 1158.
math_font_base: 230, 232, 234, 1230, 1231.
math_fraction: 1180, 1181.
math_given: 208, 413, 1046, 1151, 1154, 1222, 1223, 1224.
math_glue: 716, 732, 766.
math_group: 269, 1136, 1150, 1153, 1186.
\mathinner primitive: 1156.
math_kern: 717, 730.
math_left_group: 269, 1065, 1068, 1069, 1150, 1191.
math_left_right: 1190, 1191.
math_limit_switch: 1158, 1159.
math_node: 147, 148, 175, 183, 202, 206, 622, 651, 817, 837, 866, 879*, 881, 1147.
\mathop primitive: 1156.
\mathopen primitive: 1156.
\mathord primitive: 1156.
\mathpunct primitive: 1156.
math_quad: 700, 703, 1199.
math_radical: 1162, 1163.
\mathrel primitive: 1156.
math_shift: 207, 289, 294, 298, 347, 1090, 1137, 1138, 1193, 1197, 1206.
math_shift_group: 269, 1065, 1068, 1069, 1130, 1139, 1140, 1142, 1145, 1192, 1193, 1194, 1200.
math_shift_token: 289, 1047, 1065.
math_spacing: 764, 765.
math_style: 208, 1046, 1169, 1170, 1171.
math_surround: 247, 1196.
\mathsurround primitive: 248.
math_surround_code: 247, 248.
math_text_char: 681, 752, 753, 754, 755.
math_type: 681, 683, 687, 692, 698, 720, 722, 723, 734, 735, 737, 738, 741, 742, 749, 751, 752, 753, 754, 755, 756, 1076, 1093, 1151, 1155, 1165, 1168, 1176, 1181, 1185, 1186, 1191.
math_x_height: 700, 737, 757, 758, 759.
mathex: 701.
mathsy: 700.
mathsy_end: 700.
max_answer: 105.
max_buf_stack: 30, 31*, 331, 374, 1334.
max_char_code: 207, 303, 341, 344, 1233.
max_command: 209, 210, 211, 219, 358, 366, 368, 380, 381, 478, 782.
max_d: 726, 727, 730, 760, 761, 762.
max_dead_cycles: 236, 240, 1012.
\maxdeadcycles primitive: 238.
max_dead_cycles_code: 236, 237, 238.
max_depth: 247, 980, 987.
\maxdepth primitive: 248.
max_depth_code: 247, 248.

max_dimen: 421, 460, 641, 668, 1010, 1017, 1145, 1146, 1148.
max_group_code: 269.
max_h: 592, 593, 641, 642*, 726, 727, 730, 760, 761, 762.
max_halfword: 11*, 14, 110, 111, 113, 124, 125, 126, 131, 132, 289, 290, 424, 820, 848, 850, 982, 991, 996, 1017, 1106, 1249, 1323, 1325, 1335.
max_in_open: 11*, 14, 304, 328.
max_in_stack: 301, 321, 331, 1334.
max_internal: 209, 413, 440, 448, 455, 461.
max_nest_stack: 213, 215, 216, 1334.
max_non_prefixed_command: 208, 1211, 1270.
max_param_stack: 308, 331, 390, 1334.
max_print_line: 11*, 14, 54, 58, 61, 72, 176, 537*, 638, 1280.
max_push: 592, 593, 619, 629, 642*
max_quarterword: 11*, 110, 111, 113, 274, 797, 798, 944, 1120, 1325.
max_save_stack: 271, 272, 273, 1334.
max_selector: 54, 246, 311, 465, 470, 534, 638, 1257, 1279, 1368, 1370.
max_strings: 11*, 38, 43, 111, 517, 525, 1310, 1334.
max_v: 592, 593, 641, 642*
`\meaning` primitive: 468.
meaning_code: 468, 469, 471, 472.
med_mu_skip: 224.
`\medmuskip` primitive: 226.
med_mu_skip_code: 224, 225, 226, 766.
mem: 11*, 12, 115, 116, 118, 124, 126, 131, 133, 134, 135, 140, 141, 150, 151, 157, 159, 162, 163, 164, 165, 167, 172, 182, 186, 203, 205, 206, 221, 224, 275, 291, 387, 420, 489, 605, 652, 680, 681, 683, 686, 687, 720, 725, 742, 753, 769, 770, 772, 797, 816*, 818, 819, 822, 823, 832, 843, 844, 847, 848, 850, 860, 861, 889, 925, 1149, 1151, 1160, 1163, 1165, 1181, 1186, 1247, 1248, 1311, 1312, 1339*
mem_bot: 11*, 12, 14, 111, 116, 125, 126, 162, 164, 1307, 1308, 1311, 1312.
mem_end: 116, 118, 120, 164, 165, 167, 168, 171, 172, 174, 176, 182, 293, 1311, 1312, 1334.
mem_max: 11*, 12, 14, 110, 111, 116, 120, 124, 125, 165, 166.
mem_min: 11*, 12, 111, 116, 120, 125, 165, 166, 167, 169, 170, 171, 172, 174, 178, 182, 1249, 1312, 1334.
mem_top: 11*, 12, 14, 111, 116, 162, 164, 1249, 1307, 1308, 1312.
`Memory usage...`: 639.
memory_word: 110, 113, 114, 116, 182, 212, 218, 221, 253, 268, 271, 275, 548, 549, 800, 1305.
message: 208, 1276, 1277, 1278.
`\message` primitive: 1277.
METAFONT: 589.
mid: 546.
mid_line: 87, 303, 328, 344, 347, 352, 353, 354.
min_halfword: 11*, 110, 111, 113, 115, 230, 1027, 1323, 1325.
min_internal: 208, 413, 440, 448, 455, 461.
min_quarterword: 12, 110, 111, 112*, 113, 134, 136, 140, 185, 221, 274, 549, 550, 554, 556, 557, 566, 576, 649, 668, 685, 697, 707, 713, 714, 796, 801, 803, 808, 920, 923, 924, 943, 944, 945, 946, 958, 963, 964, 965, 994, 1012, 1323, 1324, 1325.
minimal_demerits: 833, 834, 836, 845, 855.
minimum_demerits: 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`: 503.
`Missing # inserted...`: 783.
`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`: 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_field: 212, 213, 218, 422, 800, 1244.
mode_line: 212, 213, 215, 216, 304, 804, 815, 1025.
Monperrus, Martin: 1382*.
month: 236, 241*, 536, 617, 1328.
`\month` primitive: 238.
month_code: 236, 237, 238.
months: 534, 536.
more_name: 512, 516*, 526, 531.
`\moveleft` primitive: 1071.
move_past: 619, 622, 625, 629, 631, 634.
`\moveright` primitive: 1071.
movement: 607, 609, 616.
movement_node_size: 605, 607, 615.
mskip: 208, 1046, 1057, 1058, 1059.
`\mskip` primitive: 1058.
mskip_code: 1058, 1060.
mstate: 607, 611, 612.
mtype: 4*.
mu: 447, 448, 449, 453, 455, 461, 462.
`mu`: 456.
mu_error: 408, 429, 449, 455, 461.
mu_glue: 149, 155, 191, 424, 717, 732, 1058, 1060, 1061.
mu_mult: 716, 717.
mu_skip: 224, 427.
`\muskip` primitive: 411.
mu_skip_base: 224, 227, 229, 1224, 1237.
`\muskipdef` primitive: 1222.
mu_skip_def_code: 1222, 1223, 1224.
mu_val: 410, 411, 413, 424, 427, 429, 430, 449, 451, 455, 461, 465, 1060, 1228, 1236, 1237.
mult_and_add: 105.
mult_integers: 105, 1240.
multiply: 209, 265, 266, 1210, 1235, 1236, 1240.
`\multiply` primitive: 265.
`Must increase the x`: 1303.
n: 47, 65, 66, 67, 69, 91, 94, 105, 106, 107, 152, 154, 174, 182, 225, 237, 247, 252, 292, 315, 389, 482, 498, 518, 519, 523, 578, 706, 716, 717, 791, 800, 906, 934, 944, 977, 992, 993, 994, 1012, 1079, 1119, 1138, 1211, 1275, 1338*.
name: 300, 302, 303, 304, 307, 311, 313, 314, 323, 328, 329, 331, 337, 360*, 390, 483, 537*.
name_field: 84*, 300, 302.
name_in_progress: 378, 526, 527, 528, 1258.
name_length: 26, 51, 519, 523, 525.
name_of_file: 26, 27*, 51, 519, 523, 525, 530, 1379*, 1380*.
natural: 644, 705, 715, 720, 727, 735, 737, 738, 748, 754, 756, 759, 796, 799, 806, 977, 1021, 1100, 1125, 1194, 1199, 1204.
nd: 540, 541, 560, 565, 566, 569.
ne: 540, 541, 560, 565, 566, 569.
negate: 16, 65, 103, 105, 106, 107, 430, 431, 440, 448, 461, 775.
negative: 106, 413, 430, 440, 441, 448, 461.
nest: 212, 213, 216, 217, 218, 219, 413, 422, 775, 800, 995, 1244.
nest_ptr: 213, 215, 216, 217, 218, 422, 775, 800, 995, 1017, 1023, 1091, 1100, 1145, 1200, 1244.
nest_size: 11*, 213, 216, 218, 413, 1244, 1334.
new_character: 582, 755, 915, 1117, 1123, 1124.
new_choice: 689, 1172.
new_delta_from_break_width: 844.
new_delta_to_break_width: 843.
new_disc: 145, 1035, 1117.
new_font: 1256, 1257.
new_glue: 153, 154, 715, 766, 786, 793, 795, 809, 1041, 1043, 1054, 1060, 1171.
new_graf: 1090, 1091.
new_hlist: 725, 727, 743, 748, 749, 750, 754, 756, 762, 767.
new_hyph_exceptions: 934, 1252.
new_interaction: 1264, 1265.
new_kern: 156, 705, 715, 735, 738, 739, 747, 751, 753, 755, 759, 910, 1040, 1061, 1112, 1113, 1125, 1204.
new_lig_item: 144, 911, 1040.
new_ligature: 144, 910, 1035.
new_line: 303, 331, 343, 344, 345, 347, 483, 537*.
new_line_char: 59, 236, 244.
`\newlinechar` primitive: 238.
new_line_char_code: 236, 237, 238.
new_math: 147, 1196.
new_noad: 686, 720, 742, 753, 1076, 1093, 1150, 1155, 1158, 1168, 1177, 1191.
new_null_box: 136, 706, 709, 713, 720, 747, 750, 779, 793, 809, 1018, 1054, 1091, 1093.
new_param_glue: 152, 154, 679, 778, 816*, 886, 887, 1041, 1043, 1091, 1203, 1205, 1206.
new_patterns: 960, 1252.
new_penalty: 158, 767, 816*, 890, 1054, 1103, 1203, 1205, 1206.
new_rule: 139, 463, 666, 704.

new_save_level: 274, 645, 774, 785, 791, 1025, 1063, 1099, 1117, 1119, 1136.
new_skip_param: 154, 679, 969, 1001.
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.
new_style: 688, 1171.
new_trie_op: 943, 944, 945, 965.
new_whatsit: 1349, 1350, 1354, 1376, 1377.
new_write_whatsit: 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: 59, 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.
`\noboundary` primitive: 265.
no_break_yet: 829, 836, 837.
no_expand: 210, 265, 266, 366, 367.
`\noexpand` primitive: 265.
no_expand_flag: 358, 506.
`\noindent` primitive: 1088.
no_limits: 682, 1156, 1157.
`\nolimits` primitive: 1156.
no_new_control_sequence: 256, 257, 259, 264, 365, 374, 1336.
no_print: 54, 57, 58, 75, 98.
no_shrink_error_yet: 825, 826, 827.
no_tag: 544, 569.
noad_size: 681, 686, 698, 753, 761, 1186, 1187.
node_list_display: 180, 184, 188, 190, 195, 197.
node_r_stays_active: 830, 851, 854.
node_size: 124, 126, 127, 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.
nonnegative_integer: 69, 101, 107.
nonstop_mode: 73, 86, 360;* 363, 484, 1262, 1263.
`\nonstopmode` primitive: 1262.
nop: 583, 585, 586, 588, 590.
norm_min: 1091, 1200, 1376, 1377.
normal: 135, 136, 149, 150, 153, 155, 156, 164, 177, 186, 189, 191, 305, 331, 336, 369, 439, 448, 471, 473, 480, 482, 485, 489, 490, 507, 619, 625, 629, 634, 650, 657, 658, 659, 660, 664, 665, 666, 667, 672, 673, 674, 676, 677, 678, 682, 686, 696, 716, 732, 749, 777, 801, 810, 811, 825, 826, 896, 897, 899, 976, 988, 1004, 1009, 1156, 1163, 1165, 1181, 1201, 1219, 1220, 1221, 1239.
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, 972, 973, 1138, 1146, 1365.
`notexpanded:`: 258.
np: 540, 541, 560, 565, 566, 575, 576.
nucleus: 681, 682, 683, 686, 687, 690, 696, 698, 720, 725, 734, 735, 736, 737, 738, 741, 742, 749, 750, 752, 753, 754, 755, 1076, 1093, 1150, 1151, 1155, 1158, 1163, 1165, 1168, 1186, 1191.
null: 115, 116, 118, 120, 122, 123, 125, 126, 135, 136, 144, 145, 149, 150, 151, 152, 153, 154, 164, 168, 169, 175, 176, 182, 200, 201, 202, 204, 210, 212, 218, 219, 222, 223, 232, 233, 275, 292, 295, 306, 307, 312, 314, 325, 331, 357, 358, 371, 374, 382, 383, 386, 390, 391, 392, 397, 400, 407, 410, 420, 423, 452, 464, 466, 473, 478, 482, 489, 490, 497, 505, 508, 549, 552, 576, 578, 582, 606, 611, 615, 619, 623, 629, 632, 648, 649, 651, 655, 658, 664, 666, 668, 673, 676, 681, 685, 689, 692, 715, 718, 719, 720, 721, 726, 731, 732, 752, 754, 755, 756, 760, 761, 766, 767, 771, 774, 776, 777, 783, 784, 789, 790, 791, 792, 794, 796, 797, 799, 801, 804, 805, 806, 807, 812, 821, 829, 837, 840, 846, 847, 848, 850, 856, 857, 858, 859, 863, 864, 865, 867, 869, 872, 877;* 878, 879;* 881, 882, 883, 884, 885, 887, 888, 889, 894, 896, 898, 903, 906, 907, 908, 910, 911, 913, 914, 915, 916, 917, 918, 928, 932, 935, 968, 969, 970, 972,

973, 977, 978, 979, 981, 991, 992, 993, 994, 998, 999, 1000, 1009, 1010, 1011, 1012, 1014, 1015, 1016, 1017, 1018, 1020, 1021, 1022, 1023, 1026, 1027, 1028, 1030, 1032, 1035, 1036, 1037, 1038, 1040, 1042, 1043, 1070, 1074, 1075, 1076, 1079, 1080, 1081, 1083, 1087, 1091, 1105, 1110, 1121, 1123, 1124, 1131, 1136, 1139, 1145, 1146, 1149, 1167, 1174, 1176, 1181, 1184, 1185, 1186, 1194, 1196, 1199, 1202, 1205, 1206, 1226, 1227, 1247, 1248, 1283, 1288, 1296, 1311, 1312, 1335, 1339* 1353, 1354, 1368, 1369, 1375.
null delimiter: 240, 1065.
null_character: 555, 556, 722, 723.
null_code: 22, 232.
null_cs: 222, 262, 263, 354, 374, 1257.
null_delimiter: 684, 685, 1181.
null_delimiter_space: 247, 706.
\nulldelimiterspace primitive: 248.
null_delimiter_space_code: 247, 248.
null_flag: 138, 139, 463, 653, 779, 793, 801.
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.
num: 450, 458, 585, 587, 590.
num_style: 702, 744.
Number too big: 445.
\number primitive: 468.
number_code: 468, 469, 470, 471, 472.
numerator: 683, 690, 697, 698, 744, 1181, 1185.
num1: 700, 744.
num2: 700, 744.
num3: 700, 744.
nw: 540, 541, 560, 565, 566, 569.
nx_plus_y: 105, 455, 716, 1240.
o: 264, 607, 649, 668, 791, 800.
octal_token: 438, 444.
odd: 62, 100, 193, 504, 758, 898, 902, 908, 909, 913, 914, 1211, 1218.
off_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_l: 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: 208, 265, 266, 788, 789, 1126.
\omit primitive: 265.
omit_error: 1126, 1129.
omit_template: 162, 789, 790.
Only one # is allowed...: 784.
op_byte: 545, 557, 741, 753, 909, 911, 1040.
op_noad: 682, 690, 696, 698, 726, 728, 733, 749, 761, 1156, 1157, 1159.
op_start: 920, 921, 924, 945, 1325.
open_area: 1341, 1351, 1356, 1374.
open_ext: 1341, 1351, 1356, 1374.
open_fmt_file: 524, 1337.
\openin primitive: 1272.
open_log_file: 78, 92, 360* 471, 532* 534, 535, 537* 1257, 1335.
open_name: 1341, 1351, 1356, 1374.
open_noad: 682, 690, 696, 698, 728, 733, 761, 762, 1156, 1157.
open_node: 1341, 1344, 1346, 1348, 1356, 1357, 1358, 1373.
open_node_size: 1341, 1351, 1357, 1358.
open_or_close_in: 1274, 1275.
\openout primitive: 1344.
open_parens: 304, 331, 362, 537* 1335.
\or primitive: 491.
or_code: 489, 491, 492, 500, 509.
ord: 20.
ord_noad: 681, 682, 686, 687, 690, 696, 698, 728, 729, 733, 752, 753, 761, 764, 765, 1075, 1155, 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.
other_token: 289, 405, 438, 441, 445, 464, 503, 1065, 1221.
othercases: 10*
others: 10*
otherwise: 10*
Ouch...clobbered: 1332*
out_param: 207, 289, 291, 294, 357.
out_param_token: 289, 479.
out_what: 1366, 1367, 1373, 1375.
\outer primitive: 1208.
outer_call: 210, 275, 339, 351, 353, 354, 357, 366, 387, 391, 396, 780, 1152, 1295, 1369.
outer_doing_leaders: 619, 628, 629, 637.
output: 4*
Output loop...: 1024.
Output routine didn't use...: 1028.
Output written on x: 642*
\output primitive: 230.

output_active: 421, 663, 675, 986, 989, 990, 994, 1005, 1025, 1026.
output_file_name: 532* 533, 642*
output_group: 269, 1025, 1100.
output_penalty: 236.
\outputpenalty primitive: 238.
output_penalty_code: 236, 237, 238, 1013.
output_routine: 230, 1012, 1025.
output_routine_loc: 230, 231, 232, 307, 323, 1226.
output_text: 307, 314, 323, 1025, 1026.
\over primitive: 1178.
over_code: 1178, 1179, 1182.
over_noad: 687, 690, 696, 698, 733, 761, 1156.
\overwithdelims primitive: 1178.
overbar: 705, 734, 737.
overflow: 35, 42, 43, 94, 120, 125, 216, 260, 273, 274, 321, 328, 374, 390, 517, 580, 940, 944, 954, 964, 1333*
overflow in arithmetic: 9* 104.
Overfull \hbox...: 666.
Overfull \vbox...: 677.
overfull boxes: 854.
overfull_rule: 247, 666, 800, 804.
\overfullrule primitive: 248.
overfull_rule_code: 247, 248.
\overline primitive: 1156.
p: 120, 123, 125, 130, 131, 136, 139, 144, 145, 147, 151, 152, 153, 154, 156, 158, 167, 172, 174, 176, 178, 182, 198, 200, 201, 202, 204, 218, 259, 262, 263, 276, 277, 278, 279, 281, 284, 292, 295, 306, 315, 323, 325, 336, 366, 389, 407, 413, 450, 464, 465, 473, 482, 497, 498, 582, 607, 615, 619, 629, 638, 649, 668, 679, 686, 688, 689, 691, 692, 704, 705, 709, 711, 715, 716, 717, 720, 726, 735, 738, 743, 749, 752, 756, 772, 774, 787, 791, 799, 800, 826, 906, 934, 948, 949, 953, 957, 959, 960, 966, 968, 970, 993, 994, 1012, 1064, 1068, 1075, 1079, 1086, 1093, 1101, 1105, 1110, 1113, 1119, 1123, 1138, 1151, 1155, 1160, 1174, 1176, 1184, 1191, 1194, 1211, 1236, 1244, 1288, 1293, 1302, 1303, 1348, 1349, 1355, 1368, 1370, 1373.
pack_begin_line: 661, 662, 663, 675, 804, 815.
pack_buffered_name: 523, 524, 1379*
pack_cur_name: 529, 530, 537* 1275, 1374.
pack_file_name: 519, 529, 537* 563.
pack_job_name: 529, 532* 534, 1328.
pack_lig: 1035.
package: 1085, 1086.
packed_ASCII_code: 38, 39, 947.
page: 304.
page_contents: 421, 980, 986, 987, 991, 1000, 1001, 1008.
page_depth: 982, 987, 991, 1002, 1003, 1004, 1008, 1010.
\pagedepth primitive: 983.
\pagefilstretch primitive: 983.
\pagefillstretch primitive: 983.
\pagefilllstretch primitive: 983.
page_goal: 980, 982, 986, 987, 1005, 1006, 1007, 1008, 1009, 1010.
\pagegoal primitive: 983.
page_head: 162, 215, 980, 986, 988, 991, 1014, 1017, 1023, 1026, 1054.
page_ins_head: 162, 981, 986, 1005, 1008, 1018, 1019, 1020.
page_ins_node_size: 981, 1009, 1019.
page_loc: 638, 640.
page_max_depth: 980, 982, 987, 991, 1003, 1017.
page_shrink: 982, 985, 1004, 1007, 1008, 1009.
\pageshrink primitive: 983.
page_so_far: 421, 982, 985, 987, 1004, 1007, 1009, 1245.
page_stack: 304.
\pagestretch primitive: 983.
page_tail: 215, 980, 986, 991, 998, 1000, 1017, 1023, 1026, 1054.
page_total: 982, 985, 1002, 1003, 1004, 1007, 1008, 1010.
\pagetotal primitive: 983.
panicking: 165, 166, 1031, 1339*
\par primitive: 334.
par_end: 207, 334, 335, 1046, 1094.
par_fill_skip: 224, 816*
\parfillskip primitive: 226.
par_fill_skip_code: 224, 225, 226, 816*
par_indent: 247, 1091, 1093.
\parindent primitive: 248.
par_indent_code: 247, 248.
par_loc: 333, 334, 351, 1313, 1314.
\parshape primitive: 265.
par_shape_loc: 230, 232, 233, 1070, 1248.
par_shape_ptr: 230, 232, 233, 423, 814, 847, 848, 850, 889, 1070, 1149, 1249.
par_skip: 224, 1091.
\parskip primitive: 226.
par_skip_code: 224, 225, 226, 1091.
par_token: 333, 334, 339, 392, 395, 399, 1095, 1314.
Paragraph ended before...: 396.
param: 542, 547, 558.
param_base: 550, 552, 558, 566, 574, 575, 576, 578, 580, 700, 701, 1042, 1322, 1323.
param_end: 558.
param_ptr: 308, 323, 324, 331, 390.
param_size: 11* 308, 390, 1334.

param_stack: 307, 308, 324, 359, 388, 389, 390.
param_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* 693, 764.
pass_number: 821, 845, 864.
pass_text: 366, 494, 500, 509, 510.
passive: 821, 845, 846, 864, 865.
passive_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.
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, 968, 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, 325, 333, 336, 366, 382, 388, 389, 407, 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, 689, 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, 826, 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, 1110, 1113, 1119, 1123, 1138, 1151, 1155, 1160, 1174, 1176, 1184, 1191, 1194, 1198, 1211, 1236, 1257, 1288, 1293, 1302, 1303, 1345, 1348, 1349, 1355, 1368, 1370, 1373.
Poirot, 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_ptr: 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_lig_stack: 910, 911.
pop_nest: 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.
post_display_penalty: 236, 1205, 1206.
`\postdisplaypenalty` primitive: 238.
post_display_penalty_code: 236, 237, 238.
post_line_break: 876* 877*
post_post: 585, 586, 590, 591, 642*
pre: 583, 585, 586, 617.
pre_break: 145, 175, 195, 202, 206, 858, 869, 882, 885, 915, 1117, 1119.
pre_display_penalty: 236, 1203, 1206.
`\predisplaypenalty` primitive: 238.
pre_display_penalty_code: 236, 237, 238.
pre_display_size: 247, 1138, 1145, 1148, 1203.
`\predisplaysize` primitive: 248.
pre_display_size_code: 247, 248, 1145.
preamble: 768, 774.
preamble: 770, 771, 772, 777, 786, 801, 804.
preamble of `DVI` file: 617.
precedes_break: 148, 868, 973, 1000.
prefix: 209, 1208, 1209, 1210, 1211.
prefixed_command: 1210, 1211, 1270.
prepare_mag: 288, 457, 617, 642* 1333*
pretolerance: 236, 828, 863.
`\pretolerance` primitive: 238.
pretolerance_code: 236, 237, 238.
prev_break: 821, 845, 846, 877* 878.

prev_depth: 212, 213, 215, 418, 679, 775, 786, 787, 1025, 1056, 1083, 1099, 1167, 1206, 1242, 1243.
\prevdepth primitive: 416.
prev_dp: 970, 972, 973, 974, 976.
prev_graf: 212, 213, 215, 216, 422, 814, 816* 864, 877* 890, 1091, 1149, 1200, 1242.
\prevgraf primitive: 265.
prev_p: 862* 863, 866, 867, 868, 869, 968, 969, 970, 973, 1012, 1014, 1017, 1022.
prev_prev_r: 830, 832, 843, 844, 860.
prev_r: 829, 830, 832, 843, 844, 845, 851, 854, 860.
prev_s: 862* 894, 896.
primitive: 226, 230, 238, 248, 264, 265, 266, 298, 334, 376, 384, 411, 416, 468, 487, 491, 553, 780, 983, 1052, 1058, 1071, 1088, 1107, 1114, 1141, 1156, 1169, 1178, 1188, 1208, 1219, 1222, 1230, 1250, 1254, 1262, 1272, 1277, 1286, 1291, 1331, 1332* 1344.
print: 54, 59, 60, 62, 63, 68, 70, 71, 73, 85, 86, 89, 91, 94, 95, 175, 177, 178, 182, 183, 184, 185, 186, 187, 188, 190, 191, 192, 193, 195, 211, 218, 219, 225, 233, 234, 237, 247, 251, 262, 263, 284, 288, 294, 298, 299, 306, 317, 318, 323, 336, 338, 339, 363, 373, 395, 396, 398, 400, 428, 454, 456, 459, 465, 472, 502, 509, 530, 534, 536, 561, 567, 579, 581, 617, 638, 639, 642* 660, 663, 666, 674, 675, 677, 692, 694, 697, 723, 776, 846, 856, 936, 978, 985, 986, 987, 1006, 1011, 1015, 1024, 1049, 1064, 1095, 1132, 1166, 1213, 1232, 1237, 1257, 1259, 1261, 1295, 1296, 1298, 1309, 1311, 1318, 1320, 1322, 1324, 1328, 1334, 1335, 1338* 1339* 1346, 1356.
print_ASCII: 68, 174, 176, 298, 581, 691, 723.
print_char: 58, 59, 60, 64, 65, 66, 67, 69, 70, 82, 91, 94, 95, 103, 114, 171, 172, 174, 175, 176, 177, 178, 184, 186, 187, 188, 189, 190, 191, 193, 218, 219, 223, 229, 233, 234, 235, 242, 251, 252, 255, 262, 284, 285, 294, 296, 299, 306, 313, 317, 362, 472, 509, 536, 537* 561, 581, 617, 638, 639, 642* 691, 723, 846, 856, 933, 1006, 1011, 1065, 1069, 1212, 1213, 1280, 1294, 1296, 1311, 1320, 1322, 1324, 1328, 1333* 1335, 1340, 1355, 1356.
print_cmd_chr: 223, 233, 266, 296, 298, 299, 323, 336, 418, 428, 503, 510, 1049, 1066, 1128, 1212, 1213, 1237, 1335, 1339*
print_cs: 262, 293, 314, 401.
print_current_string: 70, 182, 692.
print_delimiter: 691, 696, 697.
print_err: 72, 73, 93, 94, 95, 98, 288, 336, 338, 346, 370, 373, 395, 396, 398, 403, 408, 415, 418, 428, 433, 434, 435, 436, 437, 442, 445, 446, 454, 456, 459, 460, 475, 476, 479, 486, 500, 503, 510, 530, 561, 577, 579, 641, 723, 776, 783, 784, 792, 826, 936, 937, 960, 961, 962, 963, 976, 978, 993, 1004, 1009, 1015, 1024, 1027, 1028, 1047, 1049, 1064, 1066, 1068, 1069, 1078, 1082, 1084, 1095, 1099, 1110, 1120, 1121, 1127, 1128, 1129, 1132, 1135, 1159, 1161, 1166, 1177, 1183, 1192, 1195, 1197, 1207, 1212, 1213, 1215, 1225, 1232, 1236, 1237, 1241, 1243, 1244, 1252, 1258, 1259, 1283, 1298, 1304, 1372.
print_esc: 63, 86, 176, 184, 187, 188, 189, 190, 191, 192, 194, 195, 196, 197, 225, 227, 229, 231, 233, 234, 235, 237, 239, 242, 247, 249, 251, 262, 263, 266, 267, 292, 293, 294, 323, 335, 373, 377, 385, 412, 417, 428, 469, 486, 488, 492, 500, 579, 691, 694, 695, 696, 697, 699, 776, 781, 792, 856, 936, 960, 961, 978, 984, 986, 1009, 1015, 1028, 1053, 1059, 1065, 1069, 1072, 1089, 1095, 1099, 1108, 1115, 1120, 1129, 1132, 1135, 1143, 1157, 1166, 1179, 1189, 1192, 1209, 1213, 1220, 1223, 1231, 1241, 1244, 1251, 1255, 1263, 1273, 1278, 1287, 1292, 1295, 1322, 1335, 1346, 1355, 1356.
print_fam_and_char: 691, 692, 696.
print_file_name: 518, 530, 561, 1322, 1356.
print_font_and_char: 176, 183, 193.
print_glue: 177, 178, 185, 186.
print_hex: 67, 691, 1223.
print_int: 65, 91, 94, 103, 114, 168, 169, 170, 171, 172, 185, 188, 194, 195, 218, 219, 227, 229, 231, 233, 234, 235, 239, 242, 249, 251, 255, 285, 288, 313, 336, 400, 465, 472, 509, 536, 561, 579, 617, 638, 639, 642* 660, 663, 667, 674, 675, 678, 691, 723, 846, 856, 933, 986, 1006, 1009, 1011, 1024, 1028, 1099, 1232, 1296, 1309, 1311, 1318, 1320, 1324, 1328, 1335, 1339* 1355, 1356.
print_length_param: 247, 249, 251.
print_ln: 57, 58, 59, 61, 62, 71, 86, 89, 90, 114, 182, 198, 218, 236, 245, 296, 306, 314, 317, 330, 360* 363, 401, 484, 534, 537* 638, 639, 660, 663, 666, 667, 674, 675, 677, 678, 692, 986, 1265, 1280, 1309, 1311, 1318, 1320, 1324, 1340, 1370.
print_locs: 167.
print_mark: 176, 196, 1356.
print_meaning: 296, 472, 1294.
print_mode: 211, 218, 299, 1049.
print_nl: 62, 73, 82, 85, 90, 168, 169, 170, 171, 172, 218, 219, 245, 255, 285, 288, 299, 306, 311, 313, 314, 323, 360* 400, 530, 534, 581, 638, 639, 641, 642* 660, 666, 667, 674, 677, 678, 846, 856, 857, 863, 933, 986, 987, 992, 1006, 1011, 1121, 1294, 1296, 1297, 1322, 1324, 1328, 1333* 1335, 1338* 1370.
print_param: 237, 239, 242.

print_plus: 985.
print_plus_end: 985.
print_roman_int: 69, 472.
print_rule_dimen: 176, 187.
print_scaled: 103, 114, 176, 177, 178, 184, 188, 191, 192, 219, 251, 465, 472, 561, 666, 677, 697, 985, 986, 987, 1006, 1011, 1259, 1261, 1322.
print_size: 699, 723, 1231.
print_skip_param: 189, 225, 227, 229.
print_spec: 178, 188, 189, 190, 229, 465.
print_style: 690, 694, 1170.
print_subsidiary_data: 692, 696, 697.
print_the_digs: 64, 65, 67.
print_totals: 218, 985, 986, 1006.
print_two: 66, 536, 617.
print_word: 114, 1339*.
print_write_whatsit: 1355, 1356.
printed_node: 821, 856, 857, 858, 864.
privileged: 1051, 1054, 1130, 1140.
prompt_file_name: 530, 532*, 535, 537*, 1328, 1374.
prompt_input: 71, 83, 87, 360*, 363, 484, 530.
prune_movements: 615, 619, 629.
prune_page_top: 968, 977, 1021.
pseudo: 54, 57, 58, 59, 316.
pstack: 388, 390, 396, 400.
pt: 453.
punct_noad: 682, 690, 696, 698, 728, 752, 761, 1156, 1157.
push: 584, 585, 586, 590, 592, 601, 608, 616, 619, 629.
push_alignment: 772, 774.
push_input: 321, 323, 325, 328.
push_math: 1136, 1139, 1145, 1153, 1172, 1174, 1191.
push_nest: 216, 774, 786, 787, 1025, 1083, 1091, 1099, 1117, 1119, 1136, 1167, 1200.
put: 26, 29, 1305.
put_rule: 585, 586, 633.
put1: 585.
put2: 585.
put3: 585.
put4: 585.
q: 123, 125, 130, 131, 144, 151, 152, 153, 167, 172, 202, 204, 218, 275, 292, 315, 336, 366, 389, 407, 450, 461, 463, 464, 465, 473, 482, 497, 498, 607, 649, 705, 706, 709, 712, 720, 726, 734, 735, 736, 737, 738, 743, 749, 752, 756, 762, 791, 800, 826, 830, 862*, 877*, 901, 906, 934, 948, 953, 957, 959, 960, 968, 970, 994, 1012, 1043, 1068, 1079, 1093, 1105, 1119, 1123, 1138, 1184, 1198, 1211, 1236, 1302, 1303, 1348, 1370.
qi: 112* 545, 549, 564, 570, 573, 576, 582, 620, 753, 907, 908, 911, 913, 923, 958, 959, 981, 1008, 1009, 1034, 1035, 1038, 1039, 1040, 1100, 1151, 1155, 1160, 1165, 1309, 1325.
qo: 112* 159, 174, 176, 185, 188, 554, 570, 576, 602, 620, 691, 708, 722, 723, 741, 752, 755, 896, 897, 898, 903, 909, 923, 945, 981, 986, 1008, 1018, 1021, 1039, 1310, 1324, 1325.
qqqq: 110, 113, 114, 550, 554, 569, 573, 574, 683, 713, 741, 752, 909, 1039, 1181, 1305, 1306.
quad: 547, 558, 1146.
quad_code: 547, 558.
quarterword: 110, 113, 144, 253, 264, 271, 276, 277, 279, 281, 298, 300, 323, 592, 681, 706, 709, 711, 712, 724, 738, 749, 877*, 921, 943, 944, 947, 960, 1061, 1079, 1105.
qw: 560, 564, 570, 573, 576.
r: 108, 123, 125, 131, 204, 218, 366, 389, 465, 482, 498, 649, 668, 706, 720, 726, 752, 791, 800, 829, 862*, 877*, 901, 953, 966, 970, 994, 1012, 1123, 1160, 1198, 1236, 1348, 1370.
r_count: 912, 914, 918.
r_hyf: 891, 892, 894, 899, 902, 923, 1362.
r_type: 726, 727, 728, 729, 760, 766, 767.
radical: 208, 265, 266, 1046, 1162.
\radical primitive: 265.
radical_noad: 683, 690, 696, 698, 733, 761, 1163.
radical_noad_size: 683, 698, 761, 1163.
radix: 366, 438, 439, 440, 444, 445, 448.
radix_backup: 366.
\raise primitive: 1071.
Ramshaw, Lyle Harold: 539.
rbrace_ptr: 389, 399, 400.
read: 52, 53, 1338*, 1339*.
\read primitive: 265.
read_file: 480, 485, 486, 1275.
read_font_info: 560, 564, 1040, 1257.
read_ln: 31* 52.
read_open: 480, 481, 483, 485, 486, 501, 1275.
read_sixteen: 564, 565, 568.
read_to_cs: 209, 265, 266, 1210, 1225.
read_toks: 303, 482, 1225.
ready_already: 1331, 1332*.
real: 3, 109* 110, 182, 186, 619, 629, 1123, 1125.
real addition: 1125.
real division: 658, 664, 673, 676, 810, 811, 1123, 1125.
real multiplication: 114, 186, 625, 634, 809, 1125.
rebox: 715, 744, 750.
reconstitute: 905, 906, 913, 915, 916, 917, 1032.
recursion: 76, 78, 173, 180, 198, 202, 203, 366, 402, 407, 498, 527, 592, 618, 692, 719, 720,

725, 754, 949, 957, 959, 1333,* 1375.
ref_count: 389, 390, 401.
reference counts: 150, 200, 201, 203, 275, 291, 307.
register: 209, 411, 412, 413, 1210, 1235, 1236, 1237.
rel_noad: 682, 690, 696, 698, 728, 761, 767, 1156, 1157.
rel_penalty: 236, 682, 761.
\relpenalty primitive: 238.
rel_penalty_code: 236, 237, 238.
relax: 207, 265, 266, 358, 372, 404, 506, 1045, 1224.
\relax primitive: 265.
rem_byte: 545, 554, 557, 570, 708, 713, 740, 749, 753, 911, 1040.
remainder: 104, 106, 107, 457, 458, 543, 544, 545, 716, 717.
remove_item: 208, 1104, 1107, 1108.
rep: 546.
replace_count: 145, 175, 195, 840, 858, 869, 882, 883, 918, 1081, 1105, 1120.
report_illegal_case: 1045, 1050, 1051, 1243, 1377.
reset: 26, 27,* 31.*
reset_OK: 27.*
restart: 15, 125, 126, 341, 346, 357, 359, 360,* 362, 380, 752, 753, 782, 785, 789, 1151, 1215.
restore_old_value: 268, 276, 282.
restore_trace: 283, 284.
restore_zero: 268, 276, 278.
result: 45, 46.
resume_after_display: 800, 1199, 1200, 1206.
reswitch: 15, 341, 343, 352, 463, 619, 620, 649, 651, 652, 726, 728, 934, 935, 1029, 1030, 1036, 1045, 1138, 1147, 1151.
return: 15, 16.
rewrite: 26, 27.*
rewrite_OK: 27.*
rh: 110, 113, 114, 118, 213, 219, 221, 234, 256, 268, 685, 921, 958.
\right primitive: 1188.
right_brace: 207, 289, 294, 298, 347, 357, 389, 442, 474, 477, 785, 935, 961, 1067, 1252.
right_brace_limit: 289, 325, 392, 399, 400, 474, 477.
right_brace_token: 289, 339, 1065, 1127, 1226, 1371.
right_delimiter: 683, 697, 748, 1181, 1182.
right_hyphen_min: 236, 1091, 1200, 1376, 1377.
\righthyphenmin primitive: 238.
right_hyphen_min_code: 236, 237, 238.
right_noad: 687, 690, 696, 698, 725, 728, 760, 761, 762, 1184, 1188, 1191.
right_ptr: 605, 606, 607, 615.
right_skip: 224, 827, 880, 881.
\rightskip primitive: 226.
right_skip_code: 224, 225, 226, 881, 886.
right1: 585, 586, 607, 610, 616.
right2: 585, 610.
right3: 585, 610.
right4: 585, 610.
Rivera, Luis: 1382.*
rlink: 124, 125, 126, 127, 129, 130, 131, 132, 145, 149, 164, 169, 772, 819, 821, 1311, 1312.
\romannumeral primitive: 468.
roman_numeral_code: 468, 469, 471, 472.
round: 3, 114, 186, 625, 634, 809, 1125.
round_decimals: 102, 103, 452.
rover: 124, 125, 126, 127, 128, 129, 130, 131, 132, 164, 169, 1311, 1312.
rt_hit: 906, 907, 910, 911, 1033, 1035, 1040.
rule_dp: 592, 622, 624, 626, 631, 633, 635.
rule_ht: 592, 622, 624, 626, 631, 633, 634, 635, 636.
rule_node: 138, 139, 148, 175, 183, 202, 206, 622, 626, 631, 635, 651, 653, 669, 670, 730, 761, 805, 841, 842, 866, 870, 871, 968, 973, 1000, 1074, 1087, 1121, 1147.
rule_node_size: 138, 139, 202, 206.
rule_save: 800, 804.
rule_wd: 592, 622, 624, 625, 626, 627, 631, 633, 635.
rules aligning with characters: 589.
runaway: 120, 306, 338, 396, 486.
Runaway...: 306.
s: 45, 46, 58, 59, 60, 62, 63, 93, 94, 95, 103, 108, 125, 130, 147, 177, 178, 264, 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, 987, 1012, 1060, 1061, 1123, 1138, 1198, 1236, 1257, 1279, 1349, 1355.
save_cond_ptr: 498, 500, 509.
save_cs_ptr: 774, 777.
save_cur_val: 450, 455.
save_for_after: 280, 1271.
save_h: 619, 623, 627, 628, 629, 632, 637.
save_index: 268, 274, 276, 280, 282.
save_level: 268, 269, 274, 276, 280, 282.
save_link: 830, 857.
save_loc: 619, 629.
save_ptr: 268, 271, 272, 273, 274, 276, 280, 282, 283, 285, 645, 804, 1086, 1099, 1100, 1117, 1120, 1142, 1153, 1168, 1172, 1174, 1186, 1194, 1304.
save_scanner_status: 366, 369, 389, 470, 471, 494, 498, 507.
save_size: 11,* 111, 271, 273, 1334.
save_split_top_skip: 1012, 1014.

save_stack: 203, 268, 270, 271, 273, 274, 275, 276, 277, 281, 282, 283, 285, 300, 372, 489, 645, 768, 1062, 1071, 1131, 1140, 1150, 1153, 1339.*
save_style: 720, 726, 754.
save_type: 268, 274, 276, 280, 282.
save_v: 619, 623, 628, 629, 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.
sc: 110, 113, 114, 135, 150, 159, 164, 213, 219, 247, 250, 251, 413, 420, 425, 550, 552, 554, 557, 558, 571, 573, 575, 580, 700, 701, 775, 822, 823, 832, 843, 844, 848, 850, 860, 861, 889, 1042, 1149, 1206, 1247, 1248, 1253.
scaled: 101, 102, 103, 104, 105, 106, 107, 108, 110, 113, 147, 150, 156, 176, 177, 447, 448, 450, 453, 548, 549, 560, 584, 592, 607, 616, 619, 629, 646, 649, 668, 679, 704, 705, 706, 712, 715, 716, 717, 719, 726, 735, 736, 737, 738, 743, 749, 756, 762, 791, 800, 823, 830, 839, 847, 877,* 906, 970, 971, 977, 980, 982, 994, 1012, 1068, 1086, 1123, 1138, 1198, 1257.
`scaled`: 1258.
scaled_base: 247, 249, 251, 1224, 1237.
scan_box: 1073, 1084, 1241.
scan_char_num: 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_eight_bit_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,* 1257, 1275, 1351.
scan_font_ident: 415, 426, 471, 577, 578, 1234, 1253.
scan_four_bit_int: 435, 501, 577, 1234, 1275, 1350.
scan_glue: 410, 461, 782, 1060, 1228, 1238.
scan_int: 409, 410, 432, 433, 434, 435, 436, 437, 438, 440, 447, 448, 461, 471, 503, 504, 509, 578, 1103, 1225, 1228, 1232, 1238, 1240, 1243, 1244, 1246, 1248, 1253, 1258, 1350, 1377.
scan_keyword: 162, 407, 453, 454, 455, 456, 458, 462, 463, 645, 1082, 1225, 1236, 1258.
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_normal_dimen: 448, 463, 503, 645, 1073, 1082, 1182, 1183, 1228, 1238, 1243, 1245, 1247, 1248, 1253, 1259.
scan_optional_equals: 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_toks: 291, 464, 473, 960, 1101, 1218, 1226, 1279, 1288, 1352, 1354, 1371.
scan_twenty_seven_bit_int: 437, 1151, 1154, 1160.
scanned_result: 413, 414, 415, 418, 422, 425, 426, 428.
scanned_result_end: 413.
scanner_status: 305, 306, 331, 336, 339, 366, 369, 389, 391, 470, 471, 473, 482, 494, 498, 507, 777, 789.
`\scriptfont` primitive: 1230.
script_mlist: 689, 695, 698, 731, 1174.
`\scriptscriptfont` primitive: 1230.
script_script_mlist: 689, 695, 698, 731, 1174.
script_script_size: 699, 756, 1195, 1230.
script_script_style: 688, 694, 731, 1169.
`\scriptscriptstyle` primitive: 1169.
script_size: 699, 756, 1195, 1230.
script_space: 247, 757, 758, 759.
`\scriptspace` primitive: 248.
script_space_code: 247, 248.
script_style: 688, 694, 702, 703, 731, 756, 762, 766, 1169.
`\scriptstyle` primitive: 1169.
scripts_allowed: 687, 1176.
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_pass: 828, 863, 866.
second_width: 847, 848, 849, 850, 889.
Sedgewick, Robert: 2.*
`see the transcript file...`: 1335.
selector: 54, 55, 57, 58, 59, 62, 71, 75, 86, 90, 92, 98, 245, 311, 312, 316, 360,* 465, 470, 534, 535, 617, 638, 1257, 1265, 1279, 1298, 1328, 1333,* 1335, 1368, 1370.
semi_simple_group: 269, 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_allowed: 76, 77, 1241, 1270.
set_box_dimen: 209, 413, 416, 417, 1210, 1242.

set_break_width_to_background: 837.
set_char_0: 585, 586, 620.
set_conversion: 458.
set_conversion_end: 458.
set_cur_lang: 934, 960, 1091, 1200.
set_cur_r: 908, 910, 911.
set_font: 209, 413, 553, 577, 1210, 1217, 1257, 1261.
set_glue_ratio_one: 109,* 664, 676, 810, 811.
set_glue_ratio_zero: 109,* 136, 657, 658, 664, 672, 673, 676, 810, 811.
set_height_zero: 970.
set_interaction: 209, 1210, 1262, 1263, 1264.
set_interrupt: 4,* 1381,* 1382.*
`\setlanguage` primitive: 1344.
set_language_code: 1344, 1346, 1348.
set_math_char: 1154, 1155.
set_page_dimen: 209, 413, 982, 983, 984, 1210, 1242.
set_page_int: 209, 413, 416, 417, 1210, 1242.
set_page_so_far_zero: 987.
set_prev_graf: 209, 265, 266, 413, 1210, 1242.
set_rule: 583, 585, 586, 624.
set_shape: 209, 265, 266, 413, 1210, 1248.
set_trick_count: 316, 317, 318, 320.
set1: 585, 586, 620.
set2: 585.
set3: 585.
set4: 585.
sf_code: 230, 232, 1034.
`\sfcode` primitive: 1230.
sf_code_base: 230, 235, 1230, 1231, 1233.
shape_ref: 210, 232, 275, 1070, 1248.
shift_amount: 135, 136, 159, 184, 623, 628, 632, 637, 649, 653, 668, 670, 681, 706, 720, 737, 738, 749, 750, 756, 757, 759, 799, 806, 807, 808, 889, 1076, 1081, 1125, 1146, 1203, 1204, 1205.
shift_case: 1285, 1288.
shift_down: 743, 744, 745, 746, 747, 749, 751, 756, 757, 759.
shift_up: 743, 744, 745, 746, 747, 749, 751, 756, 758, 759.
ship_out: 211, 592, 638, 644, 1023, 1075.
`\shipout` primitive: 1071.
ship_out_flag: 1071, 1075.
short_display: 173, 174, 175, 193, 663, 857, 1339.*
short_real: 109,* 110.
shortcut: 447, 448.
shortfall: 830, 851, 852, 853.
shorthand_def: 209, 1210, 1222, 1223, 1224.
`\show` primitive: 1291.
show_activities: 218, 1293.
show_box: 180, 182, 198, 218, 219, 236, 638, 641, 663, 675, 986, 992, 1121, 1296, 1339.*
`\showbox` primitive: 1291.
show_box_breadth: 236, 1339.*
`\showboxbreadth` primitive: 238.
show_box_breadth_code: 236, 237, 238.
show_box_code: 1291, 1292, 1293.
show_box_depth: 236, 1339.*
`\showboxdepth` primitive: 238.
show_box_depth_code: 236, 237, 238.
show_code: 1291, 1293.
show_context: 54, 78, 82, 88, 310, 311, 318, 530, 535, 537.*
show_cur_cmd_chr: 299, 367, 1031.
show_eqtb: 252, 284.
show_info: 692, 693.
show_lists: 1291, 1292, 1293.
`\showlists` primitive: 1291.
show_node_list: 173, 176, 180, 181, 182, 195, 198, 233, 690, 692, 693, 695, 1339.*
`\showthe` primitive: 1291.
show_the_code: 1291, 1292.
show_token_list: 176, 223, 233, 292, 295, 306, 319, 320, 400, 1339,* 1368.
show_whatever: 1290, 1293.
shown_mode: 213, 215, 299.
shrink: 150, 151, 164, 178, 431, 462, 625, 634, 656, 671, 716, 809, 825, 827, 838, 868, 976, 1004, 1009, 1042, 1044, 1148, 1229, 1239, 1240.
shrink_order: 150, 164, 178, 462, 625, 634, 656, 671, 716, 809, 825, 826, 976, 1004, 1009, 1148, 1239.
shrinking: 135, 186, 619, 629, 664, 676, 809, 810, 811, 1148.
si: 38, 42, 69, 951, 964, 1310.
signal: 4,* 1381.*
simple_group: 269, 1063, 1068.
Single-character primitives: 267.
  `\-`: 1114.
  `\/`: 265.
  `\␣`: 265.
single_base: 222, 262, 263, 264, 354, 374, 442, 1257, 1289.
skew_char: 426, 549, 552, 576, 741, 1253, 1322, 1323.
`\skewchar` primitive: 1254.
skip: 224, 427, 1009.
`\skip` primitive: 411.
skip_base: 224, 227, 229, 1224, 1237.
skip_blanks: 303, 344, 345, 347, 349, 354.
skip_byte: 545, 557, 741, 752, 753, 909, 1039.
skip_code: 1058, 1059, 1060.

\skipdef primitive: 1222.
skip_def_code: 1222, 1223, 1224.
skip_line: 336, 493, 494.
skipping: 305, 306, 336, 494.
slant: 547, 558, 575, 1123, 1125.
slant_code: 547, 558.
slow_print: 60, 61, 63, 518, 536, 537,* 581, 642,* 1261, 1280, 1283, 1328, 1333,* 1339.*
small_char: 683, 691, 697, 706, 1160.
small_fam: 683, 691, 697, 706, 1160.
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.
small_number: 101, 102, 147, 152, 154, 264, 366, 389, 413, 438, 440, 450, 461, 470, 482, 489, 494, 497, 498, 523, 607, 649, 668, 688, 706, 719, 720, 726, 756, 762, 829, 892, 893, 905, 906, 921, 934, 944, 960, 970, 987, 1060, 1086, 1091, 1176, 1181, 1191, 1198, 1211, 1236, 1247, 1257, 1325, 1335, 1349, 1350, 1370, 1373.
so: 38, 45, 59, 60, 69, 70, 264, 407, 464, 519, 603, 617, 766, 931, 953, 955, 956, 959, 963, 1309, 1368.
Sorry, I can't find...: 524.
sort_avail: 131, 1311.
sp: 104, 587.
sp: 458.
space: 547, 558, 752, 755, 1042.
space_code: 547, 558, 578, 1042.
space_factor: 212, 213, 418, 786, 787, 799, 1030, 1034, 1043, 1044, 1056, 1076, 1083, 1091, 1093, 1117, 1119, 1123, 1196, 1200, 1242, 1243.
\spacefactor primitive: 416.
space_shrink: 547, 558, 1042.
space_shrink_code: 547, 558, 578.
space_skip: 224, 1041, 1043.
\spaceskip primitive: 226.
space_skip_code: 224, 225, 226, 1041.
space_stretch: 547, 558, 1042.
space_stretch_code: 547, 558.
space_token: 289, 393, 464, 1215.
spacer: 207, 208, 232, 289, 291, 294, 298, 303, 337, 345, 347, 348, 349, 354, 404, 406, 407, 443, 444, 452, 464, 783, 935, 961, 1030, 1045, 1221.
\span primitive: 780.
span_code: 780, 781, 782, 789, 791.
span_count: 136, 159, 185, 796, 801, 808.
span_node_size: 797, 798, 803.
spec_code: 645.
\special primitive: 1344.
special_node: 1341, 1344, 1346, 1348, 1354, 1356, 1357, 1358, 1373.
special_out: 1368, 1373.
split: 1011.
split_bot_mark: 382, 383, 977, 979.
\splitbotmark primitive: 384.
split_bot_mark_code: 382, 384, 385, 1335.
split_first_mark: 382, 383, 977, 979.
\splitfirstmark primitive: 384.
split_first_mark_code: 382, 384, 385.
split_max_depth: 140, 247, 977, 1068, 1100.
\splitmaxdepth primitive: 248.
split_max_depth_code: 247, 248.
split_top_ptr: 140, 188, 202, 206, 1021, 1022, 1100.
split_top_skip: 140, 224, 968, 977, 1012, 1014, 1021, 1100.
\splittopskip primitive: 226.
split_top_skip_code: 224, 225, 226, 969.
split_up: 981, 986, 1008, 1010, 1020, 1021.
spotless: 76, 77, 245, 1332,* 1335.
spread: 645.
sprint_cs: 223, 263, 338, 395, 396, 398, 472, 479, 484, 561, 1294.
square roots: 737.
ss_code: 1058, 1059, 1060.
ss_glue: 162, 164, 715, 1060.
stack conventions: 300.
stack_into_box: 711, 713.
stack_size: 11,* 301, 310, 321, 1334.
start: 300, 302, 303, 307, 318, 319, 323, 324, 325, 328, 329, 331, 360,* 362, 363, 369, 483, 538.
start_cs: 341, 354, 355.
start_editor: 1332,* 1380.*
start_eq_no: 1140, 1142.
start_field: 300, 302.
start_font_error_message: 561, 567.
start_here: 5, 1332.*
start_input: 366, 376, 378, 537,* 1337.
start_of_TEX: 6, 1332.*
start_par: 208, 1088, 1089, 1090, 1092.
stat: 7,* 117, 120, 121, 122, 123, 125, 130, 252, 260, 283, 284, 639, 829, 845, 855, 863, 987, 1005, 1010, 1333.*
state: 87, 300, 302, 303, 307, 311, 312, 323, 325, 328, 330, 331, 337, 341, 343, 344, 346, 347, 349, 352, 353, 354, 390, 483, 537,* 1335.
state_field: 300, 302, 1131.
stomach: 402.
stop: 207, 1045, 1046, 1052, 1053, 1054, 1094.
stop_flag: 545, 557, 741, 752, 753, 909, 1039.
store_background: 864.
store_break_width: 843.
store_fmt_file: 1302, 1335.
store_four_quarters: 564, 568, 569, 573, 574.

store_new_token: 371, 372, 393, 397, 399, 407, 464, 466, 473, 474, 476, 477, 482, 483.
store_scaled: 571, 573, 575.
str_eq_buf: 45, 259.
str_eq_str: 46, 1260.
str_number: 38, 39, 43, 45, 46, 47, 62, 63, 79* 93, 94, 95, 177, 178, 264, 284, 407, 512, 519, 525, 527, 529, 530, 532* 549, 560, 926, 929, 934, 1257, 1279, 1299, 1355.
str_pool: 38, 39, 42, 43, 45, 46, 47, 59, 60, 69, 70, 256, 260, 264, 303, 407, 464, 519, 602, 603, 617, 638, 764, 766, 929, 931, 934, 941, 1309, 1310, 1334, 1368, 1380*
str_ptr: 38, 39, 41, 43, 44, 47, 59, 60, 70, 260, 262, 517, 525, 617, 1260, 1309, 1310, 1323, 1325, 1327, 1332* 1334, 1368.
str_room: 42, 180, 260, 464, 516* 525, 939, 1257, 1279, 1328, 1333* 1368.
str_start: 38, 39, 40, 41, 43, 44, 45, 46, 47, 59, 60, 69, 70, 256, 260, 264, 407, 517, 519, 603, 617, 765, 929, 931, 934, 941, 1309, 1310, 1368, 1380*
str_toks: 464, 465, 470.
stretch: 150, 151, 164, 178, 431, 462, 625, 634, 656, 671, 716, 809, 827, 838, 868, 976, 1004, 1009, 1042, 1044, 1148, 1229, 1239, 1240.
stretch_order: 150, 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, 688, 690, 698, 730, 731, 761, 1169.
style_node_size: 688, 689, 698, 763.
sub_box: 681, 687, 692, 698, 720, 734, 735, 737, 738, 749, 754, 1076, 1093, 1168.
sub_drop: 700, 756.
sub_mark: 207, 294, 298, 347, 1046, 1175.
sub_mlist: 681, 683, 692, 720, 742, 754, 1181, 1185, 1186, 1191.
sub_style: 702, 750, 757, 759.
sub_sup: 1175, 1176.
subscr: 681, 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.
subtype: 133, 134, 135, 136, 139, 140, 143, 144, 145, 146, 147, 149, 150, 152, 153, 154, 155, 156, 158, 159, 188, 189, 190, 191, 192, 193, 424, 489, 495, 496, 625, 627, 634, 636, 649, 656, 668, 671, 681, 682, 686, 688, 689, 690, 696, 717, 730, 731, 732, 733, 749, 763, 766, 768, 786, 795, 809, 819, 820, 822, 837, 843, 844, 866, 868, 879* 881, 896, 897, 898, 899, 903, 910, 981, 986, 988, 1008, 1009, 1018, 1020, 1021, 1035, 1060, 1061, 1078, 1100, 1101, 1113, 1125, 1148, 1159, 1163, 1165, 1171, 1181, 1335, 1341, 1349, 1356, 1357, 1358, 1362, 1373, 1374.
sub1: 700, 757.
sub2: 700, 759.
succumb: 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.
supscr: 681, 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.
switch: 341, 343, 344, 346, 350.
synch_h: 616, 620, 624, 628, 633, 637, 1368.
synch_v: 616, 620, 624, 628, 632, 633, 637, 1368.
system dependencies: 2* 3, 4* 9* 10* 11* 12, 19, 21, 23, 26, 27* 28* 33* 34* 35, 36* 38, 49, 56, 59, 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, 1331, 1332* 1333* 1338* 1340.
s1: 82, 88.
s2: 82, 88.
s3: 82, 88.
s4: 82, 88.
t: 46, 107, 108, 125, 218, 241* 277, 279, 280, 281, 323, 341, 366, 389, 464, 473, 704, 705, 726, 756, 800, 830, 877* 906, 934, 966, 970, 1030, 1123, 1176, 1191, 1198, 1257, 1288.
t_open_in: 33* 37*
t_open_out: 33* 1332*
tab_mark: 207, 289, 294, 342, 347, 780, 781, 782, 783, 784, 788, 1126.
tab_skip: 224.
`\tabskip` primitive: 226.
tab_skip_code: 224, 225, 226, 778, 782, 786, 795, 809.
tab_token: 289, 1128.
tag: 543, 544, 554.

tail: 212, 213, 214, 215, 216, 424, 679, 718, 776, 786, 795, 796, 799, 812, 816,* 888, 890, 995, 1017, 1023, 1026, 1034, 1035, 1036, 1037, 1040, 1041, 1043, 1054, 1060, 1061, 1076, 1078, 1080, 1081, 1091, 1096, 1100, 1101, 1105, 1110, 1113, 1117, 1119, 1120, 1123, 1125, 1145, 1150, 1155, 1158, 1159, 1163, 1165, 1168, 1171, 1174, 1176, 1177, 1181, 1184, 1186, 1187, 1191, 1196, 1205, 1206, 1349, 1350, 1351, 1352, 1353, 1354, 1375, 1376, 1377.
tail_append: 214, 786, 795, 816,* 1035, 1037, 1040, 1054, 1056, 1060, 1061, 1091, 1093, 1100, 1103, 1112, 1113, 1117, 1150, 1158, 1163, 1165, 1168, 1171, 1172, 1177, 1191, 1196, 1203, 1205, 1206.
tail_field: 212, 213, 995.
tally: 54, 55, 57, 58, 292, 312, 315, 316, 317.
tats: 7.*
temp_head: 162, 306, 391, 396, 400, 464, 466, 467, 470, 478, 719, 720, 754, 760, 816,* 862,* 863, 864, 877,* 879,* 880, 881, 887, 968, 1064, 1065, 1194, 1196, 1199, 1297.
temp_ptr: 115, 154, 618, 619, 623, 628, 629, 632, 637, 640, 679, 692, 693, 969, 1001, 1021, 1037, 1041, 1335.
term_and_log: 54, 57, 58, 71, 75, 92, 245, 534, 1298, 1328, 1335, 1370.
term_in: 4,* 32,* 36,* 37,* 71, 1338,* 1339.*
term_input: 71, 78.
term_offset: 54, 55, 57, 58, 61, 62, 71, 537,* 638, 1280, 1333.*
term_only: 54, 55, 57, 58, 71, 75, 92, 535, 1298, 1333,* 1335.
term_out: 4,* 32,* 35, 36,* 37,* 51, 56.
terminal_input: 304, 313, 328, 330, 360.*
test_char: 906, 909.
TEX: 4.*
`TeX capacity exceeded ...`: 94.
  buffer size: 35, 328, 374.
  exception dictionary: 940.
  font memory: 580.
  grouping levels: 274.
  hash size: 260.
  input stack size: 321.
  main memory size: 120, 125.
  number of strings: 43, 517.
  parameter stack size: 390.
  pattern memory: 954, 964.
  pool size: 42.
  save size: 273.
  semantic nest size: 216.
  text input levels: 328.
`TEX.POOL check sum...`: 53.
`TEX.POOL doesn't match`: 53.
`TEX.POOL has no check sum`: 52.
`TEX.POOL line doesn't...`: 52.
TEX_area: 514,* 537.*
TEX_font_area: 514,* 563.
TEX_format_default: 520, 521,* 523.
tex_interrupt: 96.*
The TEXbook: 1, 23, 49, 108, 207, 415, 446, 456, 459, 683, 688, 764, 1215, 1331.
`TeXfonts`: 514.*
`TeXformats`: 11,* 521.*
`TeXinputs`: 514.*
`texput`: 35, 534, 1257.
text: 25,* 256, 257, 258, 259, 260, 262, 263, 264, 265, 491, 553, 780, 1188, 1216, 1257, 1318, 1369.
`Text line contains...`: 346.
text_char: 19, 20, 47.
`\textfont` primitive: 1230.
text_mlist: 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.
`\textstyle` primitive: 1169.
TEX82: 1, 99.
`TFM` files: 539.
tfm_file: 539, 560, 563, 564, 575.
`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.
then: 4.*
thick_mu_skip: 224.
`\thickmuskip` primitive: 226.
thick_mu_skip_code: 224, 225, 226, 766.
thickness: 683, 697, 725, 743, 744, 746, 747, 1182.
thin_mu_skip: 224.
`\thinmuskip` primitive: 226.
thin_mu_skip_code: 224, 225, 226, 229, 766.
`This can't happen`: 95.
  align: 800.
  copying: 206.
  curlevel: 281.
  disc1: 841.
  disc2: 842.
  disc3: 870.
  disc4: 871.
  display: 1200.
  endv: 791.
  ext1: 1348.
  ext2: 1357.

ext3: 1358.
ext4: 1373.
flushing: 202.
if: 497.
line breaking: 877*.
mlist1: 728.
mlist2: 754.
mlist3: 761.
mlist4: 766.
page: 1000.
paragraph: 866.
prefix: 1211.
pruning: 968.
right: 1185.
rightbrace: 1068.
vcenter: 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: 828, 851, 854, 863.
`Tight \hbox...`: 667.
`Tight \vbox...`: 678.
tight_fit: 817, 819, 830, 833, 834, 836, 853.
time: 236, 241*, 536, 617.
`\time` primitive: 238.
time_code: 236, 237, 238.
tini: 8.
`to`: 645, 1082, 1225.
tok_val: 410, 415, 418, 428, 465.
token: 289.
token_list: 307, 311, 312, 323, 325, 330, 337, 341, 346, 390, 1131, 1335.
token_ref_count: 200, 203, 291, 473, 482, 979.
token_show: 295, 296, 323, 401, 1279, 1284, 1297, 1370.
token_type: 307, 311, 312, 314, 319, 323, 324, 325, 327, 379, 390, 1026, 1095.
toks: 230.
`\toks` primitive: 265.
toks_base: 230, 231, 232, 233, 415, 1224, 1226, 1227.
`\toksdef` primitive: 1222.
toks_def_code: 1222, 1224.
toks_register: 209, 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: 226.
top_skip_code: 224, 225, 226, 1001.
total_demerits: 819, 845, 846, 855, 864, 874, 875.
`total height`: 986.
total_mathex_params: 701, 1195.
total_mathsy_params: 700, 1195.
total_pages: 592, 593, 617, 640, 642*.
total_shrink: 646, 650, 656, 664, 665, 666, 667, 671, 676, 677, 678, 796, 1201.
total_stretch: 646, 650, 656, 658, 659, 660, 671, 673, 674, 796.
Trabb Pardo, Luis Isidoro: 2*.
tracing_commands: 236, 367, 498, 509, 1031.
`\tracingcommands` primitive: 238.
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: 238.
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, 845, 855, 863.
`\tracingparagraphs` primitive: 238.
tracing_paragraphs_code: 236, 237, 238.
tracing_restores: 236, 283.
`\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_buf: 54, 58, 315, 317.
trick_count: 54, 58, 315, 316, 317.

Trickey, Howard Wellington: 2*.
trie: 920, 921, 922, 950, 952, 953, 954, 958, 959, 966, 1324, 1325.
trie_back: 950, 954, 956.
trie_c: 947, 948, 951, 953, 955, 956, 959, 963, 964.
trie_char: 920, 921, 923, 958, 959.
trie_fix: 958, 959.
trie_hash: 947, 948, 949, 950, 952.
trie_l: 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_o: 947, 948, 959, 963, 964.
trie_op: 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,* 921, 943, 944, 946, 1324, 1325.
trie_op_val: 943, 944, 945, 952.
trie_pack: 957, 966.
trie_pointer: 920, 921, 922, 947, 948, 949, 950, 953, 957, 959, 960, 966.
trie_ptr: 947, 951, 952, 964.
trie_r: 947, 948, 949, 955, 956, 957, 959, 963, 964.
trie_ref: 950, 952, 953, 956, 957, 959.
trie_root: 947, 949, 951, 952, 958, 966.
trie_size: 11,* 920, 948, 950, 952, 954, 964, 1325.
trie_taken: 950, 952, 953, 954, 956.
trie_used: 943, 944, 945, 946, 1324, 1325.
true: 4,* 16, 31,* 37,* 45, 46, 49, 51, 53, 71, 77, 88, 96,* 97, 98, 104, 105, 106, 107, 168, 169, 256, 257, 259, 311, 327, 328, 336, 346, 361, 362, 365, 374, 378, 407, 413, 430, 440, 444, 447, 453, 461, 462, 486, 501, 508, 512, 516,* 524, 526, 534, 563, 578, 592, 621, 628, 637, 638, 641, 663, 675, 706, 719, 791, 827, 828, 829, 851, 854, 863, 880, 882, 884, 903, 905, 910, 911, 951, 956, 962, 963, 992, 1020, 1021, 1025, 1030, 1035, 1037, 1040, 1051, 1054, 1083, 1090, 1101, 1121, 1163, 1194, 1195, 1218, 1253, 1258, 1270, 1279, 1283, 1298, 1303, 1336, 1342, 1354, 1371, 1374, 1382*.
`true`: 453.
try_break: 828, 829, 839, 851, 858, 862,* 866, 868, 869, 873, 879*.
two: 101, 102.
two_choices: 113.
two_halves: 113, 118, 124, 172, 221, 256, 684, 921, 966.
type: 4,* 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 152, 153, 155, 156, 157, 158, 159, 160, 175, 183, 184, 202, 206, 424, 489, 495, 496, 497, 505, 622, 623, 626, 628, 631, 632, 635, 637, 640, 649, 651, 653, 655, 668, 669, 670, 680, 681, 682, 683, 686, 687, 688, 689, 696, 698, 713, 715, 720, 721, 726, 727, 728, 729, 731, 732, 736, 747, 750, 752, 761, 762, 767, 768, 796, 799, 801, 805, 807, 809, 810, 811, 816,* 819, 820, 822, 830, 832, 837, 841, 842, 843, 844, 845, 856, 858, 859, 860, 861, 862,* 864, 865, 866, 868, 870, 871, 874, 875, 879,* 881, 896, 897, 899, 903, 914, 968, 970, 972, 973, 976, 978, 979, 981, 986, 988, 993, 996, 997, 1000, 1004, 1008, 1009, 1010, 1011, 1013, 1014, 1021, 1074, 1080, 1081, 1087, 1100, 1101, 1105, 1110, 1113, 1121, 1147, 1155, 1158, 1159, 1163, 1165, 1168, 1181, 1185, 1186, 1191, 1202, 1203, 1341, 1349.
`Type <return> to proceed...`: 85.
u: 69, 107, 389, 560, 706, 791, 800, 929, 934, 944, 1257.
u_close: 28,* 642*.
u_part: 768, 769, 779, 788, 794, 801.
u_template: 307, 314, 324, 788.
uc_code: 230, 232, 407.
`\uccode` primitive: 1230.
uc_code_base: 230, 235, 1230, 1231, 1286, 1288.
uc_hyph: 236, 891, 896.
`\uchyph` primitive: 238.
uc_hyph_code: 236, 237, 238.
un_hbox: 208, 1090, 1107, 1108, 1109.
`\unhbox` primitive: 1107.
`\unhcopy` primitive: 1107.
`\unkern` primitive: 1107.
`\unpenalty` primitive: 1107.
`\unskip` primitive: 1107.
un_vbox: 208, 1046, 1094, 1107, 1108, 1109.
`\unvbox` primitive: 1107.
`\unvcopy` primitive: 1107.
unbalance: 389, 391, 396, 399, 473, 477.
`Unbalanced output routine`: 1027.
`Unbalanced write...`: 1372.
`Undefined control sequence`: 370.
undefined_control_sequence: 222, 232, 256, 257, 259, 262, 268, 282, 290, 1318, 1319.
undefined_cs: 210, 222, 366, 372, 1226, 1227, 1295.
under_noad: 687, 690, 696, 698, 733, 761, 1156, 1157.
Underfull *h*box...: 816*.
`Underfull \hbox...`: 660.
`Underfull \vbox...`: 674.
`\underline` primitive: 1156.

undump: 1306, 1310, 1312, 1314, 1319, 1323, 1325, 1327.
undump_end: 1306.
undump_end_end: 1306.
undump_four_ASCII: 1310.
undump_hh: 1306, 1319, 1325.
undump_int: 1306, 1308, 1312, 1317, 1319, 1323, 1327.
undump_qqqq: 1306, 1310, 1323.
undump_size: 1306, 1310, 1321, 1325.
undump_size_end: 1306.
undump_size_end_end: 1306.
undump_wd: 1306, 1312, 1317, 1321.
unfloat: 109* 658, 664, 673, 676, 810, 811.
unhyphenated: 819, 829, 837, 864, 866, 868.
unity: 101, 103, 114, 164, 186, 453, 568, 1259.
Unix: 1332*
unpackage: 1109, 1110.
unsave: 281, 283, 791, 800, 1026, 1063, 1068, 1086, 1100, 1119, 1133, 1168, 1174, 1186, 1191, 1194, 1196, 1200.
unset_node: 136, 159, 175, 183, 184, 202, 206, 651, 669, 682, 688, 689, 768, 796, 799, 801, 805.
update_active: 861.
update_heights: 970, 972, 973, 994, 997, 1000.
update_terminal: 34* 37* 61, 71, 86, 362, 524, 537* 638, 1280, 1338*
update_width: 832, 860.
`\uppercase` primitive: 1286.
`Use of x doesn't match...`: 398.
use_err_help: 79* 80* 89, 90, 1283.
v: 69, 107, 389, 450, 706, 715, 736, 743, 749, 800, 830, 922, 934, 944, 960, 977, 1138.
v_offset: 247, 640, 641.
`\voffset` primitive: 248.
v_offset_code: 247, 248.
v_part: 768, 769, 779, 789, 794, 801.
v_template: 307, 314, 325, 390, 789, 1131.
vacuous: 440, 444, 445.
vadjust: 208, 265, 266, 1097, 1098, 1099, 1100.
`\vadjust` primitive: 265.
valign: 208, 265, 266, 1046, 1090, 1130.
`\valign` primitive: 265.
var_code: 232, 1151, 1155, 1165.
var_delimiter: 706, 737, 748, 762.
var_used: 117, 125, 130, 164, 639, 1311, 1312.
vbadness: 236, 674, 677, 678, 1012, 1017.
`\vbadness` primitive: 238.
vbadness_code: 236, 237, 238.
`\vbox` primitive: 1071.
vbox_group: 269, 1083, 1085.
vcenter: 208, 265, 266, 1046, 1167.
`\vcenter` primitive: 265.
vcenter_group: 269, 1167, 1168.
vcenter_noad: 687, 690, 696, 698, 733, 761, 1168.
vert_break: 970, 971, 976, 977, 980, 982, 1010.
very_loose_fit: 817, 819, 830, 833, 834, 836, 852.
vet_glue: 625, 634.
`\vfil` primitive: 1058.
`\vfilneg` primitive: 1058.
`\vfill` primitive: 1058.
vfuzz: 247, 677, 1012, 1017.
`\vfuzz` primitive: 248.
vfuzz_code: 247, 248.
`VIRTEX`: 1331.
virtual memory: 126.
Vitter, Jeffrey Scott: 261.
vlist_node: 137, 148, 159, 175, 183, 184, 202, 206, 505, 618, 622, 623, 628, 629, 631, 632, 637, 640, 644, 651, 668, 669, 681, 713, 715, 720, 736, 747, 750, 807, 809, 811, 841, 842, 866, 870, 871, 968, 973, 978, 1000, 1074, 1080, 1087, 1110, 1147.
vlist_out: 592, 615, 616, 618, 619, 623, 628, 629, 632, 637, 638, 640, 693, 1373.
vmode: 211, 215, 416, 417, 418, 422, 424, 501, 775, 785, 786, 804, 807, 808, 809, 812, 1025, 1029, 1045, 1046, 1048, 1056, 1057, 1071, 1072, 1073, 1076, 1078, 1079, 1080, 1083, 1090, 1091, 1094, 1098, 1099, 1103, 1105, 1109, 1110, 1111, 1130, 1167, 1243, 1244.
vmove: 208, 1048, 1071, 1072, 1073.
vpack: 236, 644, 645, 646, 668, 705, 735, 738, 759, 799, 804, 977, 1021, 1100, 1168.
vpackage: 668, 796, 977, 1017, 1086.
vrule: 208, 265, 266, 463, 1056, 1084, 1090.
`\vrule` primitive: 265.
vsize: 247, 980, 987.
`\vsize` primitive: 248.
vsize_code: 247, 248.
vskip: 208, 1046, 1057, 1058, 1059, 1078, 1094.
`\vskip` primitive: 1058.
vsplit: 967, 977, 978, 980, 1082.
`\vsplit needs a \vbox`: 978.
`\vsplit` primitive: 1071.
vsplit_code: 1071, 1072, 1079.
`\vss` primitive: 1058.
`\vtop` primitive: 1071.
vtop_code: 1071, 1072, 1083, 1085, 1086.
vtop_group: 269, 1083, 1085.
w: 114, 147, 156, 275, 278, 279, 607, 649, 668, 706, 715, 738, 791, 800, 906, 994, 1123, 1138, 1198, 1302, 1303, 1349, 1350.
w_close: 28* 1329, 1337, 1379*
w_make_name_string: 525, 1328.

w_open_in: 27,* 524, 1379.*
w_open_out: 27,* 1328.
wait: 1012, 1020, 1021, 1022.
wake_up_terminal: 34,* 37,* 51, 71, 73, 363, 484, 524, 530, 1294, 1297, 1303, 1333,* 1338.*
warning_index: 305, 331, 338, 389, 390, 395, 396, 398, 401, 473, 479, 482, 774, 777.
warning_issued: 76, 245, 1335.
was_free: 165, 167, 171.
was_hi_min: 165, 166, 167, 171.
was_lo_max: 165, 166, 167, 171.
was_mem_end: 165, 166, 167, 171.
`\wd` primitive: 416.
`WEB`: 1, 4,* 38, 40, 50, 1308.
what_lang: 1341, 1356, 1362, 1376, 1377.
what_lhm: 1341, 1356, 1362, 1376, 1377.
what_rhm: 1341, 1356, 1362, 1376, 1377.
whatsit_node: 146, 148, 175, 183, 202, 206, 622, 631, 651, 669, 730, 761, 866, 896, 899, 968, 973, 1000, 1147, 1341, 1349.
widow_penalty: 236, 1096.
`\widowpenalty` primitive: 238.
widow_penalty_code: 236, 237, 238.
`width`: 463.
width: 135, 136, 138, 139, 147, 150, 151, 155, 156, 178, 184, 187, 191, 192, 424, 429, 431, 451, 462, 463, 554, 605, 607, 611, 622, 623, 625, 626, 631, 633, 634, 635, 641, 651, 653, 656, 657, 666, 668, 669, 670, 671, 679, 683, 688, 706, 709, 714, 715, 716, 717, 731, 738, 744, 747, 749, 750, 757, 758, 759, 768, 779, 793, 796, 797, 798, 801, 802, 803, 804, 806, 807, 808, 809, 810, 811, 827, 837, 838, 841, 842, 866, 868, 870, 871, 881, 969, 976, 996, 1001, 1004, 1009, 1042, 1044, 1054, 1091, 1093, 1147, 1148, 1199, 1201, 1205, 1229, 1239, 1240.
width_base: 550, 552, 554, 566, 569, 571, 576, 1322, 1323.
width_index: 543, 550.
width_offset: 135, 416, 417, 1247.
Wirth, Niklaus: 10.*
wlog: 56, 58, 536, 1334.
wlog_cr: 56, 57, 58, 1333.*
wlog_ln: 56, 1334.
word_define: 1214, 1228, 1232, 1236.
word_file: 25,* 27,* 28,* 113, 525, 1305.
words: 204, 205, 206, 1357.
wrap_lig: 910, 911.
wrapup: 1035, 1040.
write: 37,* 56, 58.
`\write` primitive: 1344.
write_dvi: 597,* 598, 599.
write_file: 57, 58, 1342, 1374, 1378.
write_ln: 35, 37,* 51, 56, 57, 1380.*
write_loc: 1313, 1314, 1344, 1345, 1371.
write_node: 1341, 1344, 1346, 1348, 1356, 1357, 1358, 1373, 1374.
write_node_size: 1341, 1350, 1352, 1353, 1354, 1357, 1358.
write_open: 1342, 1343, 1370, 1374, 1378.
write_out: 1370, 1374.
write_stream: 1341, 1350, 1354, 1355, 1370, 1374.
write_text: 307, 314, 323, 1340, 1371.
write_tokens: 1341, 1352, 1353, 1354, 1356, 1357, 1358, 1368, 1371.
writing: 578.
wterm: 56, 58, 61.
wterm_cr: 56, 57, 58, 1333.*
wterm_ln: 56, 61, 524, 1303, 1332,* 1379.*
Wyatt, Douglas Kirk: 2.*
w0: 585, 586, 604, 609.
w1: 585, 586, 607.
w2: 585.
w3: 585.
w4: 585.
x: 100, 105, 106, 107, 587, 600, 649, 668, 706, 720, 726, 735, 737, 738, 743, 749, 756, 1123, 1302, 1303.
x_height: 547, 558, 559, 738, 1123.
x_height_code: 547, 558.
x_leaders: 149, 190, 627, 1071, 1072.
`\xleaders` primitive: 1071.
x_over_n: 106, 703, 716, 717, 986, 1008, 1009, 1010, 1240.
x_token: 364, 381, 478, 1038, 1152.
xchr: 20, 21, 23, 24, 38, 49, 58, 519, 1380.*
xclause: 16.
`\xdef` primitive: 1208.
xeq_level: 253, 254, 268, 278, 279, 283, 1304.
xn_over_d: 107, 455, 457, 458, 568, 716, 1044, 1260.
xord: 20, 24, 31,* 36,* 52, 53, 523, 525.
xpand: 473, 477, 479.
xray: 208, 1290, 1291, 1292.
xspace_skip: 224, 1043.
`\xspaceskip` primitive: 226.
xspace_skip_code: 224, 225, 226, 1043.
xxx1: 585, 586, 1368.
xxx2: 585.
xxx3: 585.
xxx4: 585, 586, 1368.
x0: 585, 586, 604, 609.
x1: 585, 586, 607.
x2: 585.
x3: 585.

x4: 585.
y: 105, 706, 726, 735, 737, 738, 743, 749, 756.
y_here: 608, 609, 611, 612, 613.
y_OK: 608, 609, 612.
y_seen: 611, 612.
year: 236, 241*, 536, 617, 1328.
`\year` primitive: 238.
year_code: 236, 237, 238.
`You already have nine...`: 476.
`You can't \insert255`: 1099.
`You can't dump...`: 1304.
`You can't use \hrule...`: 1095.
`You can't use \long...`: 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_cant: 1049, 1050, 1080, 1106.
yz_OK: 608, 609, 610, 612.
y0: 585, 586, 594, 604, 609.
y1: 585, 586, 607, 613.
y2: 585, 594.
y3: 585.
y4: 585.
z: 560, 706, 726, 743, 749, 756, 922, 927, 953, 959, 1198.
z_here: 608, 609, 611, 612, 614.
z_OK: 608, 609, 612.
z_seen: 611, 612.
Zabala Salelles, Ignacio Andrés: 2*.
zero_glue: 162, 175, 224, 228, 424, 462, 732, 802, 887, 1041, 1042, 1043, 1171, 1229.
zero_token: 445, 452, 473, 476, 479.
z0: 585, 586, 604, 609.
z1: 585, 586, 607, 614.
z2: 585.
z3: 585.
z4: 585.

⟨ Accumulate the constant until *cur_tok* is not a suitable digit 445 ⟩ Used in section 444.
⟨ Add the width of node s to *act_width* 871 ⟩ Used in section 869.
⟨ Add the width of node s to *break_width* 842 ⟩ Used in section 840.
⟨ Add the width of node s to *disc_width* 870 ⟩ Used in section 869.
⟨ Adjust for the magnification ratio 457 ⟩ Used in section 453.
⟨ Adjust for the setting of `\globaldefs` 1214 ⟩ Used in section 1211.
⟨ Adjust *shift_up* and *shift_down* for the case of a fraction line 746 ⟩ Used in section 743.
⟨ Adjust *shift_up* and *shift_down* for the case of no fraction line 745 ⟩ Used in section 743.
⟨ Advance *cur_p* to the node following the present string of characters 867 ⟩ Used in section 866.
⟨ Advance past a whatsit node in the *line_break* loop 1362 ⟩ Used in section 866.
⟨ Advance past a whatsit node in the pre-hyphenation loop 1363 ⟩ Used in section 896.
⟨ Advance r; **goto** *found* if the parameter delimiter has been fully matched, otherwise **goto** *continue* 394 ⟩ Used in section 392.
⟨ Allocate entire node p and **goto** *found* 129 ⟩ Used in section 127.
⟨ Allocate from the top of node p and **goto** *found* 128 ⟩ Used in section 127.
⟨ Apologize for inability to do the operation now, unless `\unskip` follows non-glue 1106 ⟩ Used in section 1105.
⟨ Apologize for not loading the font, **goto** *done* 567 ⟩ Used in section 566.
⟨ Append a ligature and/or kern to the translation; **goto** *continue* if the stack of inserted ligatures is nonempty 910 ⟩ Used in section 906.
⟨ Append a new leader node that uses *cur_box* 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 **goto** *big_switch* 1041 ⟩ 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 page and **goto** *contribute* 1008 ⟩ Used in section 1000.
⟨ Append any *new_hlist* entries for q, and any appropriate penalties 767 ⟩ Used in section 760.
⟨ Append box *cur_box* to the current list, shifted by *box_context* 1076 ⟩ Used in section 1075.
⟨ Append character *cur_chr* and the following characters (if any) to the current hlist in the current font; **goto** *reswitch* when a non-character has been fetched 1034 ⟩ Used in section 1030.
⟨ Append characters of $hu[j\,..\,]$ to *major_tail*, advancing j 917 ⟩ Used in section 916.
⟨ Append inter-element spacing based on *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 $p \leftarrow q$ 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* ← *show_box_depth* and *breadth_max* ← *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 *height*(v), *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, 699, 1355 ⟩ Used in section 4*.
⟨ 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 $w \leftarrow max\_dimen$ 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 legal breakpoint; on the second pass, also try to hyphenate the next word, 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 noads that can follow a *bin_noad* 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 698 ⟩ 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, 1053, 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 690 ⟩ Used in section 183.
⟨ Cases where character 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 `\halign`, $-hmode$ for `\valign` 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** 86 ⟩ Used in section 84*.
⟨ 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 569.
⟨ Check for improper alignment in displayed math 776 ⟩ Used in section 774.
⟨ Check if node p is a new champion breakpoint; then **goto** *done* if p is a forced break or if the page-so-far is already too full 974 ⟩ Used in section 972.
⟨ Check if node p is a new champion breakpoint; then if it is time for a page break, prepare for output, and either fire up the user's output routine and **return** or ship out the page and **goto** *done* 1005 ⟩ Used in section 997.

⟨ Check single-word *avail* list 168 ⟩ Used in section 167.
⟨ Check that another $ follows 1197 ⟩ Used in sections 1194, 1194, and 1206.
⟨ Check that the necessary fonts for math symbols are present; if not, flush the current math lists and set *danger* ← *true* 1195 ⟩ Used in sections 1194 and 1194.
⟨ Check that the nodes following *hb* permit hyphenation and that at least *l_hyf* + *r_hyf* letters have been found, otherwise **goto** *done1* 899 ⟩ Used in section 894.
⟨ Check the "constant" values for consistency 14, 111, 290, 522, 1249 ⟩ Used in section 1332*.
⟨ Check the pool check sum 53 ⟩ Used in section 52.
⟨ Check variable-size *avail* list 169 ⟩ Used in section 167.
⟨ Clean up the memory by removing the break nodes 865 ⟩ Used in sections 815 and 863.
⟨ Clear dimensions to zero 650 ⟩ Used in sections 649 and 668.
⟨ Clear off top level from *save_stack* 282 ⟩ Used in section 281.
⟨ Close the format file 1329 ⟩ Used in section 1302.
⟨ Coerce glue to a dimension 451 ⟩ Used in sections 449 and 455.
⟨ Compiler directives 9* ⟩ Used in section 4*.
⟨ Complain about an undefined family and set *cur_i* null 723 ⟩ Used in section 722.
⟨ Complain about an undefined macro 370 ⟩ Used in section 367.
⟨ Complain about missing `\endcsname` 373 ⟩ Used in section 372.
⟨ Complain about unknown unit and **goto** *done2* 459 ⟩ Used in section 458.
⟨ Complain that `\the` can't do this; give zero result 428 ⟩ Used in section 413.
⟨ Complain that the user should have said `\mathaccent` 1166 ⟩ Used in section 1165.
⟨ Compleat the incompleat noad 1185 ⟩ Used in section 1184.
⟨ Complete a potentially long `\show` command 1298 ⟩ Used in section 1293.
⟨ Compute result of *multiply* or *divide*, put it in *cur_val* 1240 ⟩ Used in section 1236.
⟨ Compute result of *register* or *advance*, put it in *cur_val* 1238 ⟩ Used in section 1236.
⟨ Compute the amount of skew 741 ⟩ Used in section 738.
⟨ Compute the badness, *b*, of the current page, using *awful_bad* if the box is too full 1007 ⟩ Used in section 1005.
⟨ Compute the badness, *b*, using *awful_bad* if the box is too full 975 ⟩ Used in section 974.
⟨ Compute the demerits, *d*, from *r* to *cur_p* 859 ⟩ Used in section 855.
⟨ Compute the discretionary *break_width* values 840 ⟩ Used in section 837.
⟨ Compute the hash code *h* 261 ⟩ Used in section 259.
⟨ Compute the magic offset 765 ⟩ Used in section 1337.
⟨ Compute the minimum suitable height, *w*, and the corresponding number of extension steps, *n*; also set *width*(*b*) 714 ⟩ Used in section 713.
⟨ Compute the new line width 850 ⟩ Used in section 835.
⟨ Compute the register location *l* and its type *p*; but **return** if invalid 1237 ⟩ Used in section 1236.
⟨ Compute the sum of two glue specs 1239 ⟩ Used in section 1238.
⟨ Compute the trie op code, *v*, and set *l* ← 0 965 ⟩ Used in section 963.
⟨ Compute the values of *break_width* 837 ⟩ Used in section 836.
⟨ Consider a node with matching width; **goto** *found* if it's a hit 612 ⟩ Used in section 611.
⟨ Consider the demerits for a line from *r* to *cur_p*; deactivate node *r* if it should no longer be active; then **goto** *continue* if a line from *r* to *cur_p* is infeasible, otherwise record a new feasible break 851 ⟩ Used in section 829.
⟨ Constants in the outer block 11* ⟩ Used in section 4*.
⟨ Construct a box with limits above and below it, skewed by *delta* 750 ⟩ Used in section 749.
⟨ 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 $q \leftarrow null$ 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*$(r) = qi(n)$, 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, 1200, 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, 959, 960, 966 ⟩
Used in section 942.
⟨ Declare procedures needed for displaying the elements of mlists 691, 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-related stuff 577, 578 ⟩ Used in section 409.
⟨ Declare procedures that scan restricted classes of integers 433, 434, 435, 436, 437 ⟩ Used in section 409.
⟨ Declare subprocedures for *line_break* 826, 829, 877*, 895, 942 ⟩ Used in section 815.
⟨ Declare subprocedures for *prefixed_command* 1215, 1229, 1236, 1243, 1244, 1245, 1246, 1247, 1257, 1265 ⟩
Used in section 1211.
⟨ Declare subprocedures for *var_delimiter* 709, 711, 712 ⟩ Used in section 706.
⟨ Declare the function called *fin_mlist* 1184 ⟩ Used in section 1174.
⟨ Declare the function called *open_fmt_file* 524 ⟩ Used in section 1303.
⟨ Declare the function called *reconstitute* 906 ⟩ Used in section 895.
⟨ Declare the procedure called *align_peek* 785 ⟩ Used in section 800.
⟨ Declare the procedure called *fire_up* 1012 ⟩ Used in section 994.
⟨ Declare the procedure called *get_preamble_token* 782 ⟩ Used in section 774.

⟨ Declare the procedure called *handle_right_brace* 1068 ⟩ Used in section 1030.
⟨ Declare the procedure called *init_span* 787 ⟩ Used in section 786.
⟨ Declare the procedure called *insert_relax* 379 ⟩ Used in section 366.
⟨ Declare the procedure called *macro_call* 389 ⟩ Used in section 366.
⟨ Declare the procedure called *print_cmd_chr* 298 ⟩ Used in section 252.
⟨ Declare the procedure called *print_skip_param* 225 ⟩ Used in section 179.
⟨ Declare the procedure called *restore_trace* 284 ⟩ Used in section 281.
⟨ Declare the procedure called *runaway* 306 ⟩ Used in section 119.
⟨ Declare the procedure called *show_token_list* 292 ⟩ Used in section 119.
⟨ Decry the invalid character and **goto** *restart* 346 ⟩ Used in section 344.
⟨ Delete $c -$ "0" tokens and **goto** *continue* 88 ⟩ Used in section 84*.
⟨ Delete the page-insertion nodes 1019 ⟩ Used in section 1014.
⟨ Destroy the t nodes following q, and make r point to the following node 883 ⟩ Used in section 882.
⟨ Determine horizontal glue shrink setting, then **return** or **goto** *common_ending* 664 ⟩ Used in section 657.
⟨ Determine horizontal glue stretch setting, then **return** or **goto** *common_ending* 658 ⟩ Used in section 657.
⟨ 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.
⟨ Determine the shrink order 665 ⟩ Used in sections 664, 676, and 796.
⟨ Determine the stretch order 659 ⟩ Used in sections 658, 673, and 796.
⟨ Determine the value of $height(r)$ and the appropriate glue setting; then **return** or **goto** *common_ending* 672 ⟩ Used in section 668.
⟨ Determine the value of $width(r)$ and the appropriate glue setting; then **return** or **goto** *common_ending* 657 ⟩ Used in section 649.
⟨ Determine vertical glue shrink setting, then **return** or **goto** *common_ending* 676 ⟩ Used in section 672.
⟨ Determine vertical glue stretch setting, then **return** or **goto** *common_ending* 673 ⟩ Used in section 672.
⟨ Discard erroneous prefixes and **return** 1212 ⟩ Used in section 1211.
⟨ Discard the prefixes \long and \outer if they are irrelevant 1213 ⟩ Used in section 1211.
⟨ Dispense with trivial cases of void or bad boxes 978 ⟩ Used in section 977.
⟨ Display adjustment p 197 ⟩ Used in section 183.
⟨ Display box p 184 ⟩ Used in section 183.
⟨ Display choice node p 695 ⟩ Used in section 690.
⟨ Display discretionary p 195 ⟩ Used in section 183.
⟨ Display fraction noad p 697 ⟩ Used in section 690.
⟨ Display glue p 189 ⟩ Used in section 183.
⟨ Display insertion p 188 ⟩ Used in section 183.
⟨ Display kern p 191 ⟩ Used in section 183.
⟨ Display leaders p 190 ⟩ Used in section 189.
⟨ Display ligature p 193 ⟩ Used in section 183.
⟨ Display mark p 196 ⟩ Used in section 183.
⟨ Display math node p 192 ⟩ Used in section 183.
⟨ Display node p 183 ⟩ Used in section 182.
⟨ Display normal noad p 696 ⟩ Used in section 690.
⟨ Display penalty p 194 ⟩ Used in section 183.
⟨ Display rule p 187 ⟩ Used in section 183.
⟨ Display special fields of the unset node p 185 ⟩ Used in section 184.
⟨ Display the current context 312 ⟩ Used in section 311.
⟨ Display the insertion split cost 1011 ⟩ Used in section 1010.
⟨ Display the page break cost 1006 ⟩ Used in section 1005.
⟨ Display the token (m, c) 294 ⟩ Used in section 293.
⟨ Display the value of b 502 ⟩ Used in section 498.
⟨ Display the value of $glue\_set(p)$ 186 ⟩ Used in section 184.
⟨ Display the whatsit node p 1356 ⟩ Used in section 183.

⟨ Display token p, and **return** if there are problems 293 ⟩ Used in section 292.
⟨ Do first-pass processing based on *type*(q); **goto** *done_with_noad* if a noad has been fully processed, **goto** *check_dimensions* if it has been translated into *new_hlist*(q), or **goto** *done_with_node* if a node has been fully processed 728 ⟩ Used in section 727.
⟨ Do ligature or kern command, returning to *main_lig_loop* or *main_loop_wrapup* or *main_loop_move* 1040 ⟩ Used in section 1039.
⟨ Do magic computation 320 ⟩ Used in section 292.
⟨ Do some work that has been queued up for `\write` 1374 ⟩ Used in section 1373.
⟨ Drop current token and complain that it was unmatched 1066 ⟩ Used in section 1064.
⟨ Dump a couple more things and the closing check word 1326 ⟩ Used in section 1302.
⟨ Dump constants for consistency check 1307 ⟩ Used in section 1302.
⟨ Dump regions 1 to 4 of *eqtb* 1315 ⟩ Used in section 1313.
⟨ Dump regions 5 and 6 of *eqtb* 1316 ⟩ Used in section 1313.
⟨ Dump the array info for internal font number k 1322 ⟩ Used in section 1320.
⟨ Dump the dynamic memory 1311 ⟩ Used in section 1302.
⟨ Dump the font information 1320 ⟩ Used in section 1302.
⟨ Dump the hash table 1318 ⟩ Used in section 1313.
⟨ Dump the hyphenation tables 1324 ⟩ Used in section 1302.
⟨ Dump the string pool 1309 ⟩ Used in section 1302.
⟨ Dump the table of equivalents 1313 ⟩ Used in section 1302.
⟨ Either append the insertion node p after node q, and remove it from the current page, or delete *node*(p) 1022 ⟩ Used in section 1020.
⟨ Either insert the material specified by node p into the appropriate box, or hold it for the next page; also delete node p from the current page 1020 ⟩ Used in section 1014.
⟨ Either process `\ifcase` or set b to the value of a boolean condition 501 ⟩ Used in section 498.
⟨ Empty the last bytes out of *dvi_buf* 599 ⟩ Used in section 642*.
⟨ Ensure that box 255 is empty after output 1028 ⟩ Used in section 1026.
⟨ Ensure that box 255 is empty before output 1015 ⟩ Used in section 1014.
⟨ Ensure that *trie_max* $\geq h + 256$ 954 ⟩ Used in section 953.
⟨ Enter a hyphenation exception 939 ⟩ Used in section 935.
⟨ Enter all of the patterns into a linked trie, until coming to a right brace 961 ⟩ Used in section 960.
⟨ Enter as many hyphenation exceptions as are listed, until coming to a right brace; then **return** 935 ⟩ Used in section 934.
⟨ Enter *skip_blanks* state, emit a space 349 ⟩ Used in section 347.
⟨ Error handling procedures 78, 81, 82, 93, 94, 95, 1380*, 1381* ⟩ 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 649.
⟨ 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 *link*(*def_ref*) 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; **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 420 ⟩ 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 *level* = *tok_val* 415 ⟩ Used in section 413.
⟨ Fetch an internal dimension and **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 *page_so_far* 421 ⟩ Used in section 413.
⟨ Fetch the *dead_cycles* or the *insert_penalties* 419 ⟩ Used in section 413.
⟨ Fetch the *par_shape* size 423 ⟩ Used in section 413.
⟨ Fetch the *prev_graf* 422 ⟩ Used in section 413.
⟨ Fetch the *space_factor* or the *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 *hc*, or **return** 923 ⟩ Used in section 895.
⟨ Find optimal breakpoints 863 ⟩ Used in section 815.
⟨ Find the best active node for the desired looseness 875 ⟩ Used in section 873.
⟨ Find the best way to split the insertion, and change *type*(*r*) to *split_up* 1010 ⟩ Used in section 1008.
⟨ Find the glue specification, *main_p*, for text spaces in the current font 1042 ⟩ Used in sections 1041 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 `\par` 351 ⟩ Used in section 347.
⟨ Finish line, emit a space 348 ⟩ Used in section 347.
⟨ Finish line, **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 *cur_val* if *negative* 430 ⟩ Used in section 413.
⟨ Flush the box from memory, showing statistics if requested 639 ⟩ Used in section 638.
⟨ Forbidden cases detected in *main_control* 1048, 1098, 1111, 1144 ⟩ Used in section 1045.
⟨ Generate a *down* or *right* command for *w* and **return** 610 ⟩ Used in section 607.
⟨ Generate a *y0* or *z0* command in order to reuse a previous appearance of *w* 609 ⟩ Used in section 607.
⟨ Get ready to compress the trie 952 ⟩ Used in section 966.
⟨ Get 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 improper `\hyphenation` error 936 ⟩ Used in section 935.
⟨ Global variables 13, 20, 26, 30, 39, 50, 54, 73, 76, 79*, 96*, 104, 115, 116, 117, 118, 124, 165, 173, 181, 213, 246, 253, 256, 271, 286, 297, 301, 304, 305, 308, 309, 310, 333, 361, 382, 387, 388, 410, 438, 447, 480, 489, 493, 512, 513, 520, 527, 532*, 539, 549, 550, 555, 592, 595, 605, 616, 646, 647, 661, 684, 719, 724, 764, 770, 814, 821, 823, 825, 828, 833, 839, 847, 872, 892, 900, 905, 907, 921, 926, 943, 947, 950, 971, 980, 982, 989, 1032, 1074, 1266, 1281, 1299, 1305, 1331, 1342, 1345 ⟩
Used in section 4*.
⟨ Go into display math mode 1145 ⟩ Used in section 1138.
⟨ Go into 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_i* 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* 753 ⟩ 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, **goto** *contribute* or *update_heights*; otherwise set *pi* to the penalty associated with this breakpoint 1000 ⟩ Used in section 997.
⟨ If the cursor is immediately followed by the right boundary, **goto** *reswitch*; if it's followed by an invalid character, **goto** *big_switch*; otherwise move the cursor one step to the right and **goto** *main_lig_loop* 1036 ⟩ Used in section 1034.
⟨ If the next character is a parameter number, make *cur_tok* a *match* token; but if it is a left brace, store '*left_brace*, *end_match*', set *hash_brace*, and **goto** *done* 476 ⟩ Used in section 474.
⟨ If the preamble list has been traversed, check that the row has ended 792 ⟩ Used in section 791.
⟨ If the right-hand side is a token parameter or token register, finish the assignment and **goto** *done* 1227 ⟩ Used in section 1226.
⟨ If the string *hyph_word*[h] is less than $hc[1 \mathinner{..} hn]$, **goto** *not_found*; but if the two strings are equal, set *hyf* to the hyphen positions and **goto** *found* 931 ⟩ Used in section 930.
⟨ If the string *hyph_word*[h] is less than or equal to s, interchange (*hyph_word*[h], *hyph_list*[h]) with (s, p) 941 ⟩ Used in section 940.
⟨ If there's a ligature or kern at the cursor position, update the data structures, possibly advancing j; continue until the cursor moves 909 ⟩ Used in section 906.
⟨ If there's a ligature/kern command relevant to *cur_l* and *cur_r*, adjust the text appropriately; exit to *main_loop_wrapup* 1039 ⟩ Used in section 1034.
⟨ If this font has already been loaded, set f to the internal font number and **goto** *common_ending* 1260 ⟩ Used in section 1257.
⟨ If this *sup_mark* starts an expanded character like `^^A` or `^^df`, then **goto** *reswitch*, otherwise set *state* ← *mid_line* 352 ⟩ Used in section 344.
⟨ Ignore the fraction operation and complain about this ambiguous case 1183 ⟩ Used in section 1181.
⟨ Implement `\closeout` 1353 ⟩ Used in section 1348.
⟨ Implement `\immediate` 1375 ⟩ Used in section 1348.
⟨ Implement `\openout` 1351 ⟩ Used in section 1348.
⟨ Implement `\setlanguage` 1377 ⟩ Used in section 1348.

⟨ Implement `\special` 1354 ⟩ Used in section 1348.
⟨ Implement `\write` 1352 ⟩ Used in section 1348.
⟨ Incorporate a whatsit node into a vbox 1359 ⟩ Used in section 669.
⟨ Incorporate a whatsit node into an hbox 1360 ⟩ Used in section 651.
⟨ Incorporate box dimensions into the dimensions of the hbox that will contain it 653 ⟩ Used in section 651.
⟨ Incorporate box dimensions into the dimensions of the vbox that will contain it 670 ⟩ Used in section 669.
⟨ Incorporate character dimensions into the dimensions of the hbox that will contain it, then move to the next node 654 ⟩ Used in section 651.
⟨ Incorporate glue into the horizontal totals 656 ⟩ Used in section 651.
⟨ Incorporate glue into the vertical totals 671 ⟩ Used in section 669.
⟨ Increase the number of parameters in the last font 580 ⟩ Used in section 578.
⟨ Initialize for hyphenating a paragraph 891 ⟩ Used in section 863.
⟨ Initialize table entries (done by `INITEX` only) 164, 222, 228, 232, 240, 250, 258, 552, 946, 951, 1216, 1301, 1369 ⟩ Used in section 8.
⟨ Initialize the current page, insert the `\topskip` glue ahead of p, and **goto** *continue* 1001 ⟩ Used in section 1000.
⟨ Initialize the input routines 331 ⟩ Used in section 1337.
⟨ Initialize the output routines 55, 61, 528, 533 ⟩ Used in section 1332*.
⟨ Initialize the print *selector* based on *interaction* 75 ⟩ Used in sections 1265 and 1337.
⟨ Initialize the special list heads and constant nodes 790, 797, 820, 981, 988 ⟩ Used in section 164.
⟨ Initialize variables as *ship_out* begins 617 ⟩ Used in section 640.
⟨ Initialize whatever TEX might access 8, 1382* ⟩ Used in section 4*.
⟨ Initiate or terminate input from a file 378 ⟩ Used in section 367.
⟨ Initiate the construction of an hbox or vbox, then **return** 1083 ⟩ Used in section 1079.
⟨ Input and store tokens from the next line of the file 483 ⟩ Used in section 482.
⟨ Input for `\read` from the terminal 484 ⟩ Used in section 483.
⟨ Input from external file, **goto** *restart* if no input found 343 ⟩ Used in section 341.
⟨ Input from token list, **goto** *restart* if end of list or if a parameter needs to be expanded 357 ⟩ Used in section 341.
⟨ Input the first line of *read_file*[m] 485 ⟩ Used in section 483.
⟨ Input the next line of *read_file*[m] 486 ⟩ Used in section 483.
⟨ Insert a delta node to prepare for breaks at *cur_p* 843 ⟩ Used in section 836.
⟨ Insert a delta node to prepare for the next active node 844 ⟩ Used in section 836.
⟨ Insert a dummy noad to be sub/superscripted 1177 ⟩ Used in section 1176.
⟨ Insert a new active node from *best_place*[*fit_class*] to *cur_p* 845 ⟩ Used in section 836.
⟨ Insert a new control sequence after p, then make p point to it 260 ⟩ Used in section 259.
⟨ Insert a new pattern into the linked trie 963 ⟩ Used in section 961.
⟨ Insert a new trie node between q and p, and make p point to it 964 ⟩ Used in section 963.
⟨ Insert a token containing *frozen_endv* 375 ⟩ Used in section 366.
⟨ Insert a token saved by `\afterassignment`, if any 1269 ⟩ Used in section 1211.
⟨ Insert glue for *split_top_skip* and set $p \leftarrow$ *null* 969 ⟩ Used in section 968.
⟨ Insert hyphens as specified in *hyph_list*[h] 932 ⟩ Used in section 931.
⟨ Insert macro parameter and **goto** *restart* 359 ⟩ Used in section 357.
⟨ Insert the appropriate mark text into the scanner 386 ⟩ Used in section 367.
⟨ Insert the current list into its environment 812 ⟩ Used in section 800.
⟨ Insert the pair (s, p) into the exception table 940 ⟩ Used in section 939.
⟨ Insert the $\langle v_j \rangle$ template and **goto** *restart* 789 ⟩ Used in section 342.
⟨ Insert token p into TEX's input 326 ⟩ Used in section 282.
⟨ Interpret code c and **return** if done 84* ⟩ Used in section 83.
⟨ Introduce new material from the terminal and **return** 87 ⟩ Used in section 84*.
⟨ Issue an error message if *cur_val* = *fmem_ptr* 579 ⟩ Used in section 578.

⟨ Justify the line ending at breakpoint *cur_p*, and append it to the current vertical list, together with associated penalties and other insertions 880 ⟩ Used in section 877*.
⟨ Labels in the outer block 6 ⟩ Used in section 4*.
⟨ Last-minute procedures 1333*, 1335, 1336, 1338* ⟩ Used in section 1330.
⟨ Lengthen the preamble periodically 793 ⟩ Used in section 792.
⟨ Let *cur_h* be the position of the first box, and set *leader_wd* + *lx* to the spacing between corresponding parts of boxes 627 ⟩ Used in section 626.
⟨ Let *cur_v* be the position of the first box, and set *leader_ht* + *lx* to the spacing between corresponding parts of boxes 636 ⟩ Used in section 635.
⟨ Let *d* be the natural width of node *p*; if the node is "visible," **goto** *found*; if the node is glue that stretches or shrinks, set *v* ← *max_dimen* 1147 ⟩ Used in section 1146.
⟨ Let *d* be the natural width of this glue; if stretching or shrinking, set *v* ← *max_dimen*; **goto** *found* in the case of leaders 1148 ⟩ Used in section 1147.
⟨ Let *d* be the width of the whatsit *p* 1361 ⟩ Used in section 1147.
⟨ Let *n* be the largest legal code value, based on *cur_chr* 1233 ⟩ Used in section 1232.
⟨ Link node *p* into the current page and **goto** *done* 998 ⟩ Used in section 997.
⟨ Local variables for dimension calculations 450 ⟩ Used in section 448.
⟨ Local variables for finishing a displayed formula 1198 ⟩ Used in section 1194.
⟨ Local variables for formatting calculations 315 ⟩ Used in section 311.
⟨ Local variables for hyphenation 901, 912, 922, 929 ⟩ Used in section 895.
⟨ Local variables for initialization 19, 163, 927 ⟩ Used in section 4*.
⟨ Local variables for line breaking 862*, 893 ⟩ Used in section 815.
⟨ 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 (*z*, *x*); 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*[1 .. *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_cs* 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 mlist, removing all noads 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, 651, 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* 995 ⟩ 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 1046 ⟩ 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 output 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.
⟨ Move the cursor past a pseudo-ligature, then **goto** *main_loop_lookahead* or *main_lig_loop* 1037 ⟩
Used in section 1034.
⟨ Move the data into *trie* 958 ⟩ Used in section 966.
⟨ Move to next line of file, or **goto** *restart* if there is no next line, or **return** if a `\read` line has finished 360* ⟩
Used in section 343.
⟨ Negate all three glue components of *cur_val* 431 ⟩ Used in section 430.
⟨ Nullify *width*(q) and the tabskip glue following this column 802 ⟩ Used in section 801.
⟨ Numbered cases for *debug_help* 1339* ⟩ Used in section 1338*.
⟨ Open *tfm_file* for input 563 ⟩ Used in section 562.
⟨ Other local variables for *try_break* 830 ⟩ Used in section 829.
⟨ Output a box in a vlist 632 ⟩ Used in section 631.
⟨ Output a box in an hlist 623 ⟩ Used in section 622.
⟨ Output a leader box at *cur_h*, then advance *cur_h* by *leader_wd* + *lx* 628 ⟩ Used in section 626.
⟨ Output a leader box at *cur_v*, then advance *cur_v* by *leader_ht* + *lx* 637 ⟩ Used in section 635.
⟨ Output a rule in a vlist, **goto** *next_p* 633 ⟩ Used in section 631.
⟨ Output a rule in an hlist 624 ⟩ Used in section 622.
⟨ Output leaders in a vlist, **goto** *fin_rule* if a rule or to *next_p* if done 635 ⟩ Used in section 634.
⟨ Output leaders in an hlist, **goto** *fin_rule* if a rule or to *next_p* if done 626 ⟩ Used in section 625.
⟨ Output node p for *hlist_out* and move to the next node, maintaining the condition *cur_v* = *base_line* 620 ⟩
Used in section 619.
⟨ Output node p for *vlist_out* and move to the next node, maintaining the condition *cur_h* = *left_edge* 630 ⟩
Used in section 629.
⟨ Output statistics about this job 1334 ⟩ Used in section 1333*.
⟨ Output the font definitions for all fonts that were used 643 ⟩ Used in section 642*.
⟨ Output the font name whose internal number is f 603 ⟩ Used in section 602.
⟨ Output the non-*char_node* p for *hlist_out* and move to the next node 622 ⟩ Used in section 620.
⟨ Output the non-*char_node* p for *vlist_out* 631 ⟩ Used in section 630.
⟨ Output the whatsit node p in a vlist 1366 ⟩ Used in section 631.
⟨ Output the whatsit node p in an hlist 1367 ⟩ Used in section 622.
⟨ Pack the family into *trie* relative to h 956 ⟩ Used in section 953.
⟨ Package an unset box for the current column and record its width 796 ⟩ Used in section 791.
⟨ Package the preamble list, to determine the actual tabskip glue amounts, and let p point to this prototype box 804 ⟩ Used in section 800.
⟨ Perform the default output routine 1023 ⟩ Used in section 1012.

⟨ Pontificate about improper alignment in display 1207 ⟩ Used in section 1206.
⟨ Pop the condition stack 496 ⟩ Used in sections 498, 500, 509, and 510.
⟨ Preload the default format file 1379* ⟩ Used in section 1332*.
⟨ Prepare all the boxes involved in insertions to act as queues 1018 ⟩ Used in section 1014.
⟨ Prepare to deactivate node r, and **goto** *deactivate* unless there is a reason to consider lines of text from r to *cur_p* 854 ⟩ Used in section 851.
⟨ Prepare to insert a token that matches *cur_group*, and print what it is 1065 ⟩ Used in section 1064.
⟨ Prepare to move a box or rule node to the current page, then **goto** *contribute* 1002 ⟩ Used in section 1000.
⟨ Prepare to move whatsit p to the current page, then **goto** *contribute* 1364 ⟩ Used in section 1000.
⟨ Print a short indication of the contents of node p 175 ⟩ Used in section 174.
⟨ Print a symbolic description of the new break node 846 ⟩ Used in section 845.
⟨ Print a symbolic description of this feasible break 856 ⟩ Used in section 855.
⟨ Print either '`definition`' or '`use`' or '`preamble`' or '`text`', and insert tokens that should lead to recovery 339 ⟩ Used in section 338.
⟨ Print location of current line 313 ⟩ Used in section 312.
⟨ Print newly busy locations 171 ⟩ Used in section 167.
⟨ Print string s as an error message 1283 ⟩ Used in section 1279.
⟨ Print string s on the terminal 1280 ⟩ Used in section 1279.
⟨ Print the banner line, including the date and time 536 ⟩ Used in section 534.
⟨ Print the font identifier for *font*(p) 267 ⟩ Used in sections 174 and 176.
⟨ Print the help information and **goto** *continue* 89 ⟩ Used in section 84*.
⟨ Print the list between *printed_node* and *cur_p*, then set *printed_node* ← *cur_p* 857 ⟩ Used in section 856.
⟨ Print the menu of available options 85 ⟩ Used in section 84*.
⟨ Print the result of command c 472 ⟩ Used in section 470.
⟨ Print two lines using the tricky pseudoprinted information 317 ⟩ Used in section 312.
⟨ Print type of token list 314 ⟩ Used in section 312.
⟨ Process an active-character control sequence and set *state* ← *mid_line* 353 ⟩ Used in section 344.
⟨ Process node-or-noad q as much as possible in preparation for the second pass of *mlist_to_hlist*, then move to the next item in the mlist 727 ⟩ Used in section 726.
⟨ Process whatsit p in *vert_break* loop, **goto** *not_found* 1365 ⟩ Used in section 973.
⟨ Prune the current list, if necessary, until it contains only *char_node*, *kern_node*, *hlist_node*, *vlist_node*, *rule_node*, and *ligature_node* items; set n to the length of the list, and set q to the list's tail 1121 ⟩ Used in section 1119.
⟨ Prune unwanted nodes at the beginning of the next line 879* ⟩ Used in section 877*.
⟨ Pseudoprint the line 318 ⟩ Used in section 312.
⟨ Pseudoprint the token list 319 ⟩ Used in section 312.
⟨ Push the condition stack 495 ⟩ Used in section 498.
⟨ Put each of TEX's primitives into the hash table 226, 230, 238, 248, 265, 334, 376, 384, 411, 416, 468, 487, 491, 553, 780, 983, 1052, 1058, 1071, 1088, 1107, 1114, 1141, 1156, 1169, 1178, 1188, 1208, 1219, 1222, 1230, 1250, 1254, 1262, 1272, 1277, 1286, 1291, 1344 ⟩ Used in section 1336.
⟨ Put help message on the transcript file 90 ⟩ Used in section 82.
⟨ Put the characters $hu[i+1\ ..]$ into *post_break*(r), appending to this list and to *major_tail* until synchronization has been achieved 916 ⟩ Used in section 914.
⟨ Put the characters $hu[l\ ..\ i]$ and a hyphen into *pre_break*(r) 915 ⟩ Used in section 914.
⟨ Put the fraction into a box with its delimiters, and make *new_hlist*(q) point to it 748 ⟩ Used in section 743.
⟨ Put the `\leftskip` glue at the left and detach this line 887 ⟩ Used in section 880.
⟨ Put the optimal current page into box 255, update *first_mark* and *bot_mark*, append insertions to their boxes, and put the remaining nodes back on the contribution list 1014 ⟩ Used in section 1012.
⟨ Put the (positive) 'at' size into s 1259 ⟩ Used in section 1258.
⟨ Put the `\rightskip` glue after node q 886 ⟩ Used in section 881.
⟨ Read and check the font data; *abort* if the `TFM` file is malformed; if there's no room for this font, say so and **goto** *done*; otherwise *incr*(*font_ptr*) and **goto** *done* 562 ⟩ Used in section 560.

⟨ Read box dimensions 571 ⟩ Used in section 562.
⟨ Read character data 569 ⟩ Used in section 562.
⟨ Read extensible character recipes 574 ⟩ Used in section 562.
⟨ Read font parameters 575 ⟩ Used in section 562.
⟨ Read ligature/kern program 573 ⟩ Used in section 562.
⟨ Read next line of file into *buffer*, or **goto** *restart* if the file has ended 362 ⟩ Used in section 360*.
⟨ Read one string, but return *false* if the string memory space is getting too tight for comfort 52 ⟩ Used in section 51.
⟨ Read the first line of the new file 538 ⟩ Used in section 537*.
⟨ Read the other strings from the `TEX.POOL` file and return *true*, or give an error message and return *false* 51 ⟩ Used in section 47.
⟨ Read the `TFM` header 568 ⟩ Used in section 562.
⟨ Read the `TFM` size fields 565 ⟩ Used in section 562.
⟨ Readjust the height and depth of *cur_box*, for `\vtop` 1087 ⟩ Used in section 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* 395 ⟩ Used in section 392.
⟨ Report an improper use of the macro and abort 398 ⟩ 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 677 ⟩ Used in section 676.
⟨ Report an underfull hbox and **goto** *common_ending*, if this box is sufficiently bad 660 ⟩ Used in section 658.
⟨ 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 $cur\_val \leftarrow 0$ 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 $state \leftarrow skip\_blanks$ 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 `fil` 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; **goto** *attach_sign* with *cur_val* set if found 455 ⟩ Used in section 453.
⟨ Scan preamble text until *cur_cmd* is *tab_mark* or *car_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 *c* 471 ⟩ Used in section 470.
⟨ Scan the font size specification 1258 ⟩ Used in section 1257.
⟨ Scan the parameters and make *link*(*r*) point to the macro body; but **return** if an illegal `\par` is detected 391 ⟩ Used in section 389.
⟨ Scan the preamble and record it in the *preamble* list 777 ⟩ Used in section 774.
⟨ Scan the template $\langle u_j \rangle$, putting the resulting token list in *hold_head* 783 ⟩ Used in section 779.
⟨ Scan the template $\langle v_j \rangle$, putting the resulting token list in *hold_head* 784 ⟩ Used in section 779.
⟨ Scan units and set *cur_val* to $x \cdot (cur\_val + f/2^{16})$, where there are x sp per unit; **goto** *attach_sign* if the units are internal 453 ⟩ Used in section 448.
⟨ Search *eqtb* for equivalents equal to *p* 255 ⟩ Used in section 172.
⟨ Search *hyph_list* for pointers to *p* 933 ⟩ Used in section 172.
⟨ Search *save_stack* for equivalents that point to *p* 285 ⟩ Used in section 172.
⟨ Select the appropriate case and **return** or **goto** *common_ending* 509 ⟩ Used in section 501.
⟨ Set initial values of key variables 21, 23, 24, 74, 77, 80*, 97, 166, 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 ⟩ 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.
⟨ Set the glue in node *r* and change it from an unset node 808 ⟩ Used in section 807.
⟨ Set the unset box *q* and the unset boxes in it 807 ⟩ Used in section 805.
⟨ Set the value of *b* to the badness for shrinking the line, and compute the corresponding *fit_class* 853 ⟩ Used in section 851.
⟨ Set the value of *b* to the badness for stretching the line, and compute the corresponding *fit_class* 852 ⟩ Used in section 851.
⟨ Set the value of *output_penalty* 1013 ⟩ Used in section 1012.
⟨ Set up data structures with the cursor following position *j* 908 ⟩ Used in section 906.
⟨ Set up the values of *cur_size* and *cur_mu*, based on *cur_style* 703 ⟩ Used in sections 720, 726, 730, 754, 760, and 763.
⟨ Set variable *c* to the current escape character 243 ⟩ Used in section 63.
⟨ Ship box *p* out 640 ⟩ Used in section 638.
⟨ Show equivalent *n*, in region 1 or 2 223 ⟩ Used in section 252.
⟨ Show equivalent *n*, in region 3 229 ⟩ Used in section 252.
⟨ Show equivalent *n*, in region 4 233 ⟩ Used in section 252.
⟨ Show equivalent *n*, in region 5 242 ⟩ Used in section 252.
⟨ Show equivalent *n*, in region 6 251 ⟩ Used in section 252.
⟨ Show the auxiliary field, *a* 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 **goto** *common_ending* 1294 ⟩ Used in section 1293.
⟨ Show the current value of some parameter or register, then **goto** *common_ending* 1297 ⟩ Used in section 1293.
⟨ Show the font identifier in *eqtb*[*n*] 234 ⟩ Used in section 233.
⟨ Show the halfword code in *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.
⟨ Simplify a trivial box 721 ⟩ Used in section 720.
⟨ Skip to `\else` or `\fi`, then **goto** *common_ending* 500 ⟩ Used in section 498.
⟨ Skip to node *ha*, or **goto** *done1* if no hyphenation should be attempted 896 ⟩ Used in section 894.
⟨ Skip to node *hb*, putting letters into *hu* and *hc* 897 ⟩ Used in section 894.
⟨ Sort *p* into the list starting at *rover* and advance *p* to *rlink*(*p*) 132 ⟩ Used in section 131.
⟨ Sort the hyphenation op tables into proper order 945 ⟩ Used in section 952.
⟨ Split off part of a vertical box, make *cur_box* point to it 1082 ⟩ Used in section 1079.
⟨ Squeeze the equation as much as possible; if there is an equation number that should go on a separate line by itself, set $e \leftarrow 0$ 1201 ⟩ Used in section 1199.
⟨ Start a new current page 991 ⟩ Used in sections 215 and 1017.
⟨ Store *cur_box* in a box register 1077 ⟩ Used in section 1075.
⟨ Store maximum values in the *hyf* table 924 ⟩ Used in section 923.
⟨ Store *save_stack*[*save_ptr*] in *eqtb*[*p*], unless *eqtb*[*p*] holds a global value 283 ⟩ Used in section 282.
⟨ Store the current token, but **goto** *continue* if it is a blank space that would become an undelimited parameter 393 ⟩ Used in section 392.
⟨ Subtract glue from *break_width* 838 ⟩ Used in section 837.
⟨ Subtract the width of node *v* from *break_width* 841 ⟩ Used in section 840.
⟨ Suppress expansion of the next token 369 ⟩ Used in section 367.
⟨ Swap the subscript and superscript into box *x* 742 ⟩ Used in section 738.
⟨ Switch to a larger accent if available and appropriate 740 ⟩ Used in section 738.
⟨ Tell the user what has run away and try to recover 338 ⟩ Used in section 336.
⟨ Terminate the current conditional and skip to `\fi` 510 ⟩ Used in section 367.
⟨ Test box register status 505 ⟩ Used in section 501.
⟨ Test if an integer is odd 504 ⟩ Used in section 501.
⟨ Test if two characters match 506 ⟩ Used in section 501.
⟨ Test if two macro texts match 508 ⟩ Used in section 507.
⟨ Test if two tokens match 507 ⟩ Used in section 501.
⟨ Test relation between integers or dimensions 503 ⟩ Used in section 501.
⟨ The em width for *cur_font* 558 ⟩ Used in section 455.
⟨ The x-height for *cur_font* 559 ⟩ Used in section 455.
⟨ Tidy up the parameter just scanned, and tuck it away 400 ⟩ Used in section 392.
⟨ Transfer node *p* to the adjustment list 655 ⟩ Used in section 651.
⟨ Transplant the post-break list 884 ⟩ Used in section 882.
⟨ Transplant the pre-break list 885 ⟩ Used in section 882.
⟨ Treat *cur_chr* as an active character 1152 ⟩ Used in sections 1151 and 1155.
⟨ Try the final line break at the end of the paragraph, and **goto** *done* if the desired breakpoints have been found 873 ⟩ Used in section 863.
⟨ Try to allocate within node *p* and its physical successors, and **goto** *found* if allocation was possible 127 ⟩ Used in section 125.
⟨ Try to break after a discretionary fragment, then **goto** *done5* 869 ⟩ Used in section 866.
⟨ Try to get a different log file name 535 ⟩ Used in section 534.
⟨ Try to hyphenate the following word 894 ⟩ Used in section 866.
⟨ Try to recover from mismatched `\right` 1192 ⟩ Used in section 1191.
⟨ Types in the outer block 18, 25*, 38, 101, 109*, 113, 150, 212, 269, 300, 548, 594, 920, 925 ⟩ Used in section 4*.
⟨ Undump a couple more things and the closing check word 1327 ⟩ Used in section 1303.
⟨ Undump constants for consistency check 1308 ⟩ Used in section 1303.
⟨ Undump regions 1 to 6 of *eqtb* 1317 ⟩ Used in section 1314.
⟨ Undump the array info for internal font number *k* 1323 ⟩ Used in section 1321.
⟨ Undump the dynamic memory 1312 ⟩ Used in section 1303.
⟨ Undump the font information 1321 ⟩ Used in section 1303.
⟨ Undump the hash table 1319 ⟩ Used in section 1314.

⟨ Undump the hyphenation tables 1325 ⟩ Used in section 1303.
⟨ Undump the string pool 1310 ⟩ Used in section 1303.
⟨ Undump the table of equivalents 1314 ⟩ Used in section 1303.
⟨ Update the active widths, since the first active node has been deleted 861 ⟩ Used in section 860.
⟨ Update the current height and depth measurements with respect to a glue or kern node p 976 ⟩ Used in section 972.
⟨ Update the current page measurements with respect to the glue or kern specified by node p 1004 ⟩ Used in section 997.
⟨ Update the value of *printed_node* for symbolic displays 858 ⟩ Used in section 829.
⟨ Update the values of *first_mark* and *bot_mark* 1016 ⟩ Used in section 1014.
⟨ Update the values of *last_glue*, *last_penalty*, and *last_kern* 996 ⟩ Used in section 994.
⟨ Update the values of *max_h* and *max_v*; but if the page is too large, **goto** *done* 641 ⟩ Used in section 640.
⟨ Update width entry for spanned columns 798 ⟩ Used in section 796.
⟨ Use code c to distinguish between generalized fractions 1182 ⟩ Used in section 1181.
⟨ Use node p to update the current height and depth measurements; if this node is not a legal breakpoint, **goto** *not_found* or *update_heights*, otherwise set pi to the associated penalty at the break 973 ⟩ Used in section 972.
⟨ Use size fields to allocate font information 566 ⟩ Used in section 562.
⟨ Wipe out the whatsit node p and **goto** *done* 1358 ⟩ Used in section 202.
⟨ 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 1021 ⟩ Used in section 1020.

		Section	Page
0.	About TeX$_{\text{GPC}}$	0	3
1.	Introduction	1	4
2.	The character set	17	9
3.	Input and output	25	9
4.	String handling	38	16
5.	On-line and off-line printing	54	16
6.	Reporting errors	72	16
7.	Arithmetic with scaled dimensions	99	18
8.	Packed data	110	19
9.	Dynamic memory allocation	115	20
10.	Data structures for boxes and their friends	133	20
11.	Memory layout	162	20
12.	Displaying boxes	173	20
13.	Destroying boxes	199	20
14.	Copying boxes	203	20
15.	The command codes	207	20
16.	The semantic nest	211	20
17.	The table of equivalents	220	20
18.	The hash table	256	21
19.	Saving and restoring equivalents	268	21
20.	Token lists	289	21
21.	Introduction to the syntactic routines	297	21
22.	Input stacks and states	300	21
23.	Maintaining the input stacks	321	21
24.	Getting the next token	332	21
25.	Expanding the next token	366	22
26.	Basic scanning subroutines	402	22
27.	Building token lists	464	22
28.	Conditional processing	487	22
29.	File names	511	22
30.	Font metric data	539	24
31.	Device-independent file format	583	24
32.	Shipping pages out	592	24
33.	Packaging	644	25
34.	Data structures for math mode	680	25
35.	Subroutines for math mode	699	25
36.	Typesetting math formulas	719	25
37.	Alignment	768	25
38.	Breaking paragraphs into lines	813	25
39.	Breaking paragraphs into lines, continued	862	26
40.	Pre-hyphenation	891	29
41.	Post-hyphenation	900	29
42.	Hyphenation	919	29
43.	Initializing the hyphenation tables	942	29
44.	Breaking vertical lists into pages	967	29
45.	The page builder	980	29
46.	The chief executive	1029	29
47.	Building boxes and lists	1055	29
48.	Building math lists	1136	29
49.	Mode-independent processing	1208	29
50.	Dumping and undumping the tables	1299	29
51.	The main program	1330	29
52.	Debugging	1338	31
53.	Extensions	1340	33
54.	System-dependent changes	1379	33
55.	Index	1383	35