The CTIE processor

(Version 1.1 [TEX Live])

	Section	ı Page
Introduction	1	1
Input and output	8	3
Data structures	10) 4
File I/O	19	5
Reporting errors to the user	28	8
Handling multiple change files	38	3 10
Input/output organisation	42	12
System-dependent changes	70	16
Index	79	17

Copyright © 2002, 2003 Julian Gilbey

All rights reserved.

This program is distributed WITHOUT ANY WARRANTY, express or implied.

Permission is granted to make and distribute verbatim copies of this program provided that the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this program under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

March 17, 2021 at 13:08

§1 The CTIE processor INTRODUCTION

1* Introduction. Whenever a programmer wants to change a given WEB or CWEB program (referred to as a WEB program throughout this program) because of system dependencies, she or he will create a new change file. In addition there may be a second change file to modify system independent modules of the program. But the WEB file cannot be tangled and weaved with more than one change file simultaneously. The TIE program was designed to merge a WEB file and several change files producing a new WEB file, and since the input files are tied together, the program was called TIE. Furthermore, the program could be used to merge several change files giving a new single change file. This method seems to be more important because it doesn't modify the original source file.

However, the introduction of CWEB has meant that TIE is not quite able to perform its task correctly any longer: CWEB introduced the idea of include files, which are input into CWEB files using the @i command, and TIE is unable to handle such constructs if the change files modify lines included in those files. The present program, CTIE, is designed to overcome this lack. Like TIE, upon which it is based, it can either output a single master WEB file or a master change file. However, in both cases, any include commands will be totally expanded and the files included in the output rather than the @i commands being left; this makes this code feasible, which it would not necessarily be otherwise. Other than this difference, CTIE should function identically to TIE on files which do not involve any CWEB include commands.

The algorithm used is essentially the same as that of TIE, with modifications to check for and handle @i commands. Thus, as with TIE, the method used only needs one buffer line for each input file. Thus the storage requirement of CTIE does not depend on the sizes of the input files but only on their number.

The program is written in C and has few system dependencies.

The "banner line" defined here should be changed whenever CTIE is modified. We also keep the version number here separately for ease; it is used below.

```
#define version_number "1.1"
#define banner "This is CTIE, Version 1.1"
#define copyright "Copyright 2002, 2003 Julian Gilbey."
              "| | All | rights | reserved . | | There | is | no | warranty . \n"
              "Run, with, the --version, option, for other, important, information."
     And this is the structure of the main function: this is where CTIE starts, and where it ends.
\langle The main function 3^* \rangle \equiv
  int main(int argc, string *argv)
  {
     (Set up PROGNAME feature and initialise the search path mechanism 71*)
     \langle \text{Initialise parameters } 17 \rangle
     Scan the parameters 61
     ⟨ Print the banners 60*⟩
      Get the master file started 40^*
     (Prepare the change files 41*)
     \langle \text{ Prepare the output file } 38^* \rangle
     (Process the input 57)
     (Check that all changes have been read 58)
     exit(wrap_up());
This code is used in section 2.
```

4.* We include the additional types **boolean** and **string**. CTIE replaces the complex TIE character set handling (based on that of the original WEB system) with the standard CWEB behaviour, and so uses the **char** type for input and output.

The kpathsea library (version 3.4.5 and higher) defines the **boolean** (with the values *true* and *false*) and **string** (and **const_string**) types in <kpathsea/simpletypes.h>, so we do not actually need to define them here.

2 INTRODUCTION The CTIE processor §5

5.* We don't need to predeclare any string handling functions here, as the kpathsea headers do the right thing by including <string.h> behind the scenes.

6.* The following parameters should be sufficient for most applications of CTIE.

```
#define buf\_size 1024 \triangleright maximum length of one input line \triangleleft #define max\_file\_index 32 \triangleright we don't think that anyone needs more than 32 change files \triangleleft #define xisupper(c) (isupper((unsigned char) c) \land ((unsigned char) c < °200))
```

7.* We introduce a history variable that allows us to set a return code if the operating system can use it. First we introduce the coded values for the history. This variable must be initialised. (We do this even if the value given may be the default for variables, just to document the need for the initial value.)

```
⟨Global variables 7*⟩ ≡

typedef enum {

spotless, troublesome, fatal
} return_code;

static return_code history ← spotless;

See also sections 15*, 16, 18, 22, 39*, and 66*.

This code is used in section 2.
```

§8 The CTIE processor INPUT AND OUTPUT 3

8.* Input and output. Standard output for the user is done by writing on *stdout*. Error messages are written to *stderr*. Terminal input is not needed in this version of CTIE. *stdin*, *stdout* and *stderr* are predefined as we include the <stdio.h> definitions through the kpathsea interface.

```
\langle Global #includes 8*\rangle \equiv #include <kpathsea/kpathsea.h> This code is used in section 2.
```

9.* And we need dynamic memory allocation. This should cause no trouble in any C program. The kpathsea include files handle the definition of malloc, too.

4 DATA STRUCTURES The CTIE processor §10

10.* Data structures. The multiple primary input files (master file and change files) are treated the same way. To organise the simultaneous usage of several input files, we introduce the data type in_file_modes.

The mode search indicates that CTIE searches for a match of the input line with any line of an input file in reading mode. test is used whenever a match is found and it has to be tested if the next input lines do match also. reading describes that the lines can be read without any check for matching other lines. ignore denotes that the file cannot be used. This may happen because an error has been detected or because the end of the file has been found.

file_types is used to describe whether a file is a master file or a change file. The value *unknown* is added to this type to set an initial mode for the output file. This enables us to check whether any option was used to select the kind of output. (This would even be necessary if we would assume a default action for missing options.)

```
⟨Global types 10*⟩ ≡

typedef enum {

search, test, reading, ignore
} in_file_modes;

typedef enum {

unknown, master, chf
} file_types;

See also sections 11*, 12, 13, and 14.

This code is used in section 2.
```

11.* A variable of type **out_md_type** will tell us in what state the output change file is during processing. *normal* will be the state, when we did not yet start a change, *pre* will be set when we write the lines to be changes and *post* will indicate that the replacement lines are written.

```
⟨Global types 10*⟩ +≡
typedef enum {
normal, pre, post
} out_md_type;
```

15* Every one of the primary input files might include another file using the @i include mechanism. In turn, each of these might include other files, and so on. We allow a limited number of these files to be opened simultaneously, and we store information about the currently open include files as a linked list attached to each primary file.

```
#define max\_include\_files 20 
ightharpoonup max\_imum number of include files open simultaneously 
ightharpoonup  #define max\_file\_name\_length 1024 \langle Global variables 7* \rangle +\equiv int total\_include\_files \leftarrow 0; 
ightharpoonup  count 'em 
ightharpoonup
```

 $\S19$ The CTIE processor FILE I/O 5

19* File I/O. The basic function get_line can be used to get a line from an input file. The line is stored in the buffer part of the descriptor. The components limit and line are updated. If the end of the file is reached mode is set to ignore. On some systems it might be useful to replace tab characters by a proper number of spaces since several editors used to create change files insert tab characters into a source file not under control of the user. So it might be a problem to create a matching change file.

We define get_line to read a line from a file specified by the corresponding file descriptor. This function returns true if it is successful and false if the end of the file has been reached.

```
\langle \text{ Internal functions } 19^* \rangle \equiv
  static boolean qet_line(file_index i, boolean do_includes)
     register input_description *inp\_desc \leftarrow input\_organisation[i];
     register FILE *fp:
     if (inp\_desc \neg mode \equiv ignore) return false;
  restart:
     if (inp\_desc \neg current\_include \neq \Lambda) {
        register include_description *inc\_desc \leftarrow inp\_desc \neg current\_include;
        fp \leftarrow inc\_desc\_the\_file; \langle Get include line into buffer or goto restart if end of file 24*\rangle
     else {
        fp \leftarrow inp\_desc \neg the\_file; \langle Get line into buffer, return false if end of file 20* \rangle
     if (do_includes) (Check for @i in newly read line, goto restart if include fails 26)
     return true;
  }
See also sections 32*, 42*, 43*, 46*, 47*, 48*, and 59*.
This code is used in section 2.
```

20* Lines must fit into the buffer completely. We read all characters sequentially until an end of line is found (but do not forget to check for EOF!). Too long input lines will be truncated. This will result in a damaged output if they occur in the replacement part of a change file, or in an incomplete check if the matching part is concerned. Tab character expansion might be done here.

```
\langle Get line into buffer, return false if end of file 20^*\rangle \equiv
   {
      register int c;

    b the actual character read 
    □

      register char *k;
                                     ▶ where the next character goes <</p>
     if (feof(fp)) \langle Handle end of file and return 21\rangle
      inp\_desc \neg limit \leftarrow k \leftarrow inp\_desc \neg buffer;
                                                                  ▷ beginning of buffer <</p>
      while (k \leq inp\_desc \neg buffer\_end \land (c \leftarrow getc(fp)) \neq \texttt{EOF} \land c \neq `\n')
         if ((*(k++) \leftarrow c) \neq ' \cup ' \land c \neq ' \ ') inp\_desc \neg limit \leftarrow k;
      if (k > inp\_desc \rightarrow buffer\_end)
         if ((c \leftarrow getc(fp)) \neq \texttt{EOF} \land c \neq \texttt{`\n'}) {
            unqetc(c, fp); inp\_desc \neg loc \leftarrow inp\_desc \neg buffer; err\_print(i, "!_lInput_lline_ltoo_llong");
     if (c \equiv \text{EOF} \land inp\_desc\neg limit \equiv inp\_desc\neg buffer) \land \text{Handle end of file and return 21})
      (Increment the line number and print a progress report at certain times 23)
   }
```

This code is used in section 19*.

6 FILE I/O The CTIE processor §24

The following is very similar to the above, but for the case where we are reading from an include file. \langle Get include line into buffer or **goto** restart if end of file $24^*\rangle \equiv$ { register int c: b the actual character read
 □ register char *k: if (feof(fp)) \landle Handle end of include file and goto restart 25\rangle $inp_desc \neg limit \leftarrow k \leftarrow inp_desc \neg buffer;$ ▷ beginning of buffer <</p> while $(k \leq inp_desc \neg buffer_end \land (c \leftarrow qetc(fp)) \neq \texttt{EOF} \land c \neq `\n')$ if $((*(k++) \leftarrow c) \neq ', ', \land c \neq ', r') inp_desc \neg limit \leftarrow k;$ if $(k > inp_desc \neg buffer_end)$ if $((c \leftarrow qetc(fp)) \neq EOF \land c \neq '\n')$ { ungetc(c, fp): $inp_desc \rightarrow loc \leftarrow inp_desc \rightarrow buffer$: $err_print(i, "!_{\perp}Input_{\parallel}line_{\parallel}too_{\parallel}long")$: if $(c \equiv \text{EOF} \land inp_desc \neg limit \equiv inp_desc \neg buffer)$ (Handle end of include file and **goto** restart 25) $inc_desc \neg line ++;$ This code is used in section 19*.

 $\S27$ The CTIE processor FILE I/O 7

27.* When an @i line is found in the file, we must temporarily stop reading it and start reading from the named include file. The @i line should give a complete file name with or without double quotes. We use the KPATHSEA library (in particular, the CWEBINPUTS variable) to search for this file. The remainder of the @i line after the file name is ignored.

```
#define too_long()
              total_include_files --: free(new_inc): err_print(i,"!..Include_file_name_too_long"):
              goto restart:
\langle Try to open include file, abort push if unsuccessful, go to restart 27^*\rangle \equiv
   {
     include_description *new_inc;
     char *file_name_end;
     string fullname:
     char *k:
     new\_inc \leftarrow (include\_description *) \ malloc(sizeof(include\_description));
     if (new\_inc \equiv \Lambda) fatal\_error(i, "!\_No\_memory\_for\_new\_include\_descriptor", "");
     new\_inc \neg line \leftarrow 0; k \leftarrow new\_inc \neg file\_name\_ind \leftarrow k + max\_file\_name\_length - 1;
     if (*inv\_desc \neg loc \equiv "") {
        inp\_desc \neg loc ++;
        while (*inp\_desc \neg loc \neq "" \land k \leq file\_name\_end) *k++ \leftarrow *inp\_desc \neg loc ++;
        if (inp\_desc \neg loc \equiv inp\_desc \neg limit) k \leftarrow file\_name\_end + 1; \triangleright unmatched quote is 'too long' \triangleleft
      }
     else
        while (*inp\_desc\neg loc \neq `, |, ` \land *inp\_desc\neg loc \neq `, ` \land *inp\_desc\neg loc \neq `, ", \land k < file\_name\_end)
           *k++\leftarrow *inp\_desc \rightarrow loc ++:
     if (k > file\_name\_end) too\_long();
     *k \leftarrow \text{``\0'}; fullname \leftarrow kpse\_find\_cweb(new\_inc \rightarrow file\_name);
     if (fullname) new\_inc\_the\_file \leftarrow fopen(fullname, "r");
     if (fullname \neq \Lambda \land new\_inc \neg the\_file \neq \Lambda) {
        free(fullname); new\_inc \neg parent \leftarrow inp\_desc \neg current\_include;

    b link it in 
    □

        inp\_desc \neg current\_include \leftarrow new\_inc; goto restart;

▷ success < </p>
      total_include_files --; free(new_inc);
     if (fullname) {
        free(fullname); err\_print(i, "! \Box Cannot \Box open \Box include \Box file");
     else err_print(i, "! | Cannot | find | include | file");
     goto restart;
   }
```

This code is used in section 26.

28.* Reporting errors to the user. There may be errors if a line in a given change file does not match a line in the master file or a replacement in a previous change file. Such errors are reported to the user by saying

```
err_print(file_no, "!_Error_message");
```

where *file_no* is the number of the file which is concerned by the error. Please note that no trailing dot is supplied in the error message because it is appended by *err_print*.

```
\langle Predeclaration of functions 28*\rangle \equiv
  void err_print(file_index.const char *);
See also sections 33*, 35*, and 67*.
This code is used in section 2.
29* Here is the outline of the err print function.
\langle Error handling functions 29*\rangle \equiv
                                                             ▷ prints '.' and location of error message <</p>
  void err_print(file_index i, const char *s)
  {
                         \triangleright pointers into an appropriate buffer \triangleleft
     fprintf(stderr, *s \equiv '!', ? "\n%s" : "%s", s);
     if (i > 0) (Print error location based on input buffer 30)
     else putc('\n', stderr);
     fflush(stderr); history \leftarrow troublesome;
  }
See also section 36*.
This code is used in section 2.
```

32* Some implementations may wish to pass the *history* value to the operating system so that it can be used to govern whether or not other programs are started. Here, for instance, we pass the operating system a status of 0 if and only if only harmless messages were printed.

```
⟨ Internal functions 19*⟩ +≡
  int wrap_up(void)
  {
    ⟨ Print the job history 34⟩;
    if (history > spotless) return EXIT_FAILURE;
    else return EXIT_SUCCESS;
  }
33* Always good to prototype.
⟨ Predeclaration of functions 28*⟩ +≡
  int wrap_up(void);
```

35.* If there's a system error, we may be able to give the user more information with the *pfatal_error* function. This prints out system error information if it is available.

```
\langle \text{ Predeclaration of functions } 28^* \rangle + \equiv 
void pfatal\_error(\mathbf{const \ char} *, \mathbf{const \ char} *);
```

9

```
36* ⟨Error handling functions 29*⟩ +≡

void pfatal_error(const char *s, const char *t)
{
    char *strerr ← strerror(errno);
    fprintf(stderr, "\n%s%s", s, t);
    if (strerr) fprintf(stderr, "\u00ed(%s)\u00b1n", strerr);
    else fprintf(stderr, "\u00b1");
    history ← fatal; exit(wrap_up());
}
```

37.* The <errno.h> include file for the above comes via the kpathsea interface.

This code is used in section 3*.

38.* Handling multiple change files. In the standard version we take the name of the files from the command line. It is assumed that filenames can be used as given in the command line without changes.

First there are some sections to open all files. If a file is not accessible, the run will be aborted. Otherwise the name of the open file will be displayed.

```
\langle Prepare the output file 38* \rangle \equiv
     out\_file \leftarrow fopen(out\_name, "wb"):
     if (out\_file \equiv \Lambda) {
        pfatal_error("!_|Cannot_|open/create_|output_|file","");
  }
This code is used in section 3*.
      The name of the file and the file descriptor are stored in global variables.
\langle \text{Global variables } 7^* \rangle + \equiv
  FILE *out_file:
  string out_name;
40* For the master file we start by reading its first line into the buffer, if we could open it. We use the
kpathsea library to find the file.
\langle Get the master file started 40^* \rangle \equiv
  {
     string fullname:
     fullname \leftarrow kpse\_find\_cweb(input\_organisation[0] \neg file\_name);
     if (fullname) input_organisation[0]\neg the\_file \leftarrow fopen(fullname, "r");
     if (fullname \equiv \Lambda \vee input\_organisation[0] \neg the\_file \equiv \Lambda) {
        if (fullname) {
           pfatal_error("!_iCannot_iopen_master_ifile_i", input_organisation[0]-file_name);
        else {
          fatal\_error(-1, "! \Box Cannot \Box find \Box master \Box file \Box ", input\_organisation[0] \neg file\_name);
     }
     else free(fullname);
     printf("(\%s)\n".input\_organisation[0]\neg file\_name): input\_organisation[0]\neg type\_of\_file \leftarrow master:
     get\_line(0, true);
```

11

41.* For the change files we must skip any comment part and see whether there are any changes in it. This is done by <code>init_change_file</code>.

```
\langle Prepare the change files 41*\rangle \equiv
  {
     file_index i:
     string fullname;
     i \leftarrow 1;
     while (i < no\_ch) {
        fullname \leftarrow kpse\_find\_cweb(input\_organisation[i] \neg file\_name);
        if (fullname) input_organisation[i]\neg the\_file \leftarrow fopen(fullname, "r");
        if (fullname \equiv \Lambda \vee input\_organisation[i] \neg the\_file \equiv \Lambda) {
           if (fullname) {
              pfatal\_error("! \_Cannot\_open\_change\_file\_", input\_organisation[i] \neg file\_name);
           else {
              fatal\_error(-1, "! \Box Cannot \Box find \Box change \Box file \Box ", input\_organisation[i] \neg file\_name);
        }
        else free(fullname);
        printf("(\%s)\n", input\_organisation[i] \neg file\_name); init\_change\_file(i); i++;
   }
```

This code is used in section 3*.

```
42*
       Input/output organisation.
                                                Here's a simple function that checks if two lines are different.
\langle \text{Internal functions } 19^* \rangle + \equiv
  static boolean lines_dont_match(file_index i, file_index i)
  {
     register input_description *iptr \leftarrow input\_organisation[i], *iptr \leftarrow input\_organisation[i]:
     if (iptr \rightarrow limit - iptr \rightarrow buffer \neq iptr \rightarrow limit - iptr \rightarrow buffer) return true:
     return strncmp(iptr \rightarrow buffer, iptr \rightarrow buffer, iptr \rightarrow limit - iptr \rightarrow buffer);
  }
43* Function init_change_file(i) is used to ignore all lines of the input file with index i until the next change
module is found.
\langle \text{ Internal functions } 19^* \rangle + \equiv
  static void init_change_file(file_index i)
     register input_description *inp\_desc \leftarrow input\_organisation[i];
     char ccode:
     inp\_desc \neg limit \leftarrow inp\_desc \neg buffer: \langle Skip \text{ over comment lines; } \mathbf{return} \text{ if end of file } 44* \rangle
     (Skip to the next nonblank line; return if end of file 45)
     inp\_desc \neg dont\_match \leftarrow 0;
  }
44* While looking for a line that begins with 0x in the change file, we allow lines that begin with 0, as
long as they don't begin with Qy, Qz or Qi (which would probably mean that the change file is fouled up).
\langle Skip over comment lines; return if end of file 44^*\rangle \equiv
  while (1) {
     if (\neg qet\_line(i, false)) return;
                                                 ▷ end of file reached <</p>
     if (inp\_desc \neg limit < inp\_desc \neg buffer + 2) continue;
     if (inp\_desc\_buffer[0] \neq `@`) continue;
     ccode \leftarrow inp\_desc \neg buffer[1]:
     if (xisupper(ccode)) ccode \leftarrow tolower((unsigned char) ccode);
     if (ccode \equiv 'x') break:
     if (ccode \equiv 'y' \lor ccode \equiv 'z' \lor ccode \equiv 'i') {
        inp\_desc\_loc \leftarrow inp\_desc\_buffer + 2; \ err\_print(i, "!\_Missing\_@x_in_change_ifile");
     }
This code is used in section 43*.
46* The put_line function is used to write a line from input buffer i to the output file.
\langle \text{ Internal functions } 19^* \rangle + \equiv
  static void put_line(file_index i)
  {
     \mathbf{char} *ptr \leftarrow input\_organisation[j] \neg buffer;
     \mathbf{char} * lmt \leftarrow input\_organisation[j] \neg limit;
     while (ptr < lmt) putc(*ptr++, out\_file);
     putc('\n', out\_file);
```

```
47*
       The function e_of_ch_module returns true if the input line from file i starts with Qz.
\langle \text{Internal functions } 19^* \rangle + \equiv
  static boolean e\_of\_ch\_module(file\_index i)
  {
     register input_description *inp\_desc \leftarrow input\_organisation[i]:
     if (inp\_desc \rightarrow limit \equiv \Lambda) {
        err_print(i, "!_|Change_|file_|ended_|without_|@z"); return true;
     else if (inp\_desc \neg limit > inp\_desc \neg buffer + 2)
        if (inp\_desc \neg buffer[0] \equiv 'Q' \land (inp\_desc \neg buffer[1] \equiv 'Z' \lor inp\_desc \neg buffer[1] \equiv 'z')) return true:
     return false;
48* The function e\_of\_ch\_preamble returns true if the input line from file i starts with @v.
\langle \text{Internal functions } 19^* \rangle + \equiv
  static boolean e\_of\_ch\_preamble(file\_index i)
  {
     register input_description *inp\_desc \leftarrow input\_organisation[i];
     if (inp\_desc \neg limit \ge inp\_desc \neg buffer + 2 \land inp\_desc \neg buffer [0] \equiv , 0,)
        if (inp\_desc \neg buffer[1] \equiv 'Y' \lor inp\_desc \neg buffer[1] \equiv 'v') {
          if (inp\_desc \neg dont\_match > 0) {
              inp\_desc \neg loc \leftarrow inp\_desc \neg buffer + 2; fprintf(stderr, "\n!_{l}Hmm..._{l}%d_{l}", inp\_desc \neg dont\_match);
              err_print(i, "of_the_preceding_lines_failed_to_match");
          return true;
     return false;
  }
59. We want to tell the user about our command line options if they made a mistake. This is done by the
usage_error() function. It contains merely the necessary print statements and exits afterwards.
\langle \text{Internal functions } 19^* \rangle + \equiv
  static void usage_error(void)
     \langle \text{ Print the banners } 60^* \rangle;
     fprintf(stderr, "Usage: \_ctie\_-m|-c\_outfile\_master\_changefile(s)\n");
     fprintf(stderr, "Type_{\sqcup}ctie_{\sqcup}--help_{\sqcup}for_{\sqcup}more_{\sqcup}information "); exit(EXIT_FAILURE);
  }
60* Printing our welcome banners; we only do this if we are not asked for version or help information.
\langle \text{ Print the banners } 60^* \rangle \equiv
  printf("\%s_{1})\n", banner, kpathsea\_version\_string);
                                                                         ▷ print a "banner line" <</p>
  printf("%s\n", copyright);
                                       ▷ include the copyright notice <</p>
This code is used in sections 3^* and 59^*.
```

The CTIE processor

14

63* We have to distinguish whether this is the very first file name (which is the case if $no_c ch \equiv none$) or if the next element of input_organisation must be filled. $\langle \text{ Get a file name } 63^* \rangle \equiv$ { if $(no_ch \equiv none)$ { $out_name \leftarrow *arqv;$ } else { register input_description *inp_desc; $inp_desc \leftarrow (input_description *) \ malloc(size of (input_description))$: if $(inp_desc \equiv \Lambda)$ $fatal_error(-1, "!_1No_1memory_1for_1input_1descriptor", "");$ $inp_desc \neg mode \leftarrow search$: $inp_desc \neg line \leftarrow 0$: $inp_desc \neg type_of_file \leftarrow chf$: $inp_desc \neg limit \leftarrow inp_desc \neg buffer; inp_desc \neg buffer[0] \leftarrow `_{'}'; inp_desc \neg loc \leftarrow inp_desc \neg buffer + 1;$ $inp_desc \neg buffer_end \leftarrow inp_desc \neg buffer + buf_size - 2$: $inp_desc \neg file_name \leftarrow *argv$: $inp_desc \neg current_include \leftarrow \Lambda$; $input_organisation[no_ch] \leftarrow inp_desc$; $no_ch++:$ This code is used in section 61. Here is the usage information for --help. \langle Global variables $7^* \rangle + \equiv$ $const_string\ CTIEHELP[] \leftarrow \{"Usage: | ctie| -m| -c| outfile| master| changefile(s)".$ "LUCreateLaLnewLmasterLfileLorLchangeLfileLfromLtheLgiven", "___master__(C)WEB__file__and__changefiles.", "-m, create, a, new, master, file, from, original, (C)WEB, and, change, file(s)", "-c, | create, |a, |master, |change, |file, |for, |original, |(C)WEB, |file, |from, |changefile(s)", "--help___display_this_help_and_exit", "--version, and display, version, information, and exit", Λ ; **67*** \langle Predeclaration of functions $28^* \rangle + \equiv$ static void usage_help(void); static void print_version_and_exit(const_string, const_string); static void usage_help(void) **68*** { $const_string *message \leftarrow CTIEHELP;$ while (*message) {

fputs(*message, stdout); putchar('\n'); ++message;

putchar('\n'); exit(EXIT_SUCCESS);

```
69* static void print_version_and_exit(const_string name,const_string version)
{
    printf("%su%s\n",name,version); puts(kpathsea_version_string);
    puts("Copyright_(C)_2002,2003_Julian_Gilbey.");
    puts("Kpathsea_is_copyright_(C)_1999_Free_Software_Foundation,_Inc.");
    puts("There_is_NO_warranty.ulThis_is_free_software.");
    puts("Redistribution_of_this_software_is_covered_by_the_terms_of");
    puts("both_the_CTIE_copyright_and_the_GNU_General_Public_Licence.");
    puts("For_more_information_about_these_matters,_see_the_files");
    puts("named_COPYING_and_the_CTIE_source.");
    puts("Primary_author_of_CTIE:_Julian_Gilbey.");
    puts("Kpathsea_written_by_Karl_Berry_and_others."); exit(EXIT_SUCCESS);
}
```

70* System-dependent changes. The ctie program from the original CTIE package uses the compiletime default directory or the value of the environment variable CWEBINPUTS as an alternative place to be searched for files, if they could not be found in the current directory.

This version uses the KPATHSEA mechanism for searching files. The directories to be searched for come from three sources:

- (a) a user-set environment variable CWEBINPUTS (overridden by CWEBINPUTS_ctie);
- (b) a line in KPATHSEA configuration file texmf.cnf,e.g., CWEBINPUTS=\$TEXMFDOTDIR:\$TEXMF/texmf/cweb//or CWEBINPUTS.ctie=\$TEXMFDOTDIR:\$TEXMF/texmf/cweb//:
- (c) compile-time default directories (specified in texmf.in), i.e., \$TEXMFDOTDIR:\$TEXMF/texmf/cweb//.

#define kpse_find_cweb(name) kpse_find_file(name, kpse_cweb_format, true)

71.* The simple file searching is replaced by the 'path searching' mechanism that the KPATHSEA library provides.

We set *kpse_program_name* to 'ctie'. This means if the variable CWEBINPUTS.ctie is present in texmf.cnf (or CWEBINPUTS_ctie in the environment) its value will be used as the search path for filenames. This allows different flavors of CTIE to have different search paths.

 \langle Set up PROGNAME feature and initialise the search path mechanism $71*\rangle \equiv kpse_set_program_name(argv[0], "ctie");$

This code is used in section 3*.

72* Index.

```
The following sections were changed by the change file: 1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 15, 19, 20, 24, 27, 28, 29, 32, 33, 35, 36, 37,
    38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 59, 60, 63, 66, 67, 68, 69, 70, 71, 72,
--help: 64.
                                                         file_name: 13, 14, 27, 30, 40, 41, 63.
--version: 65.
                                                         file\_name\_end: 27.*
_idsc: 14.
                                                         file_no: 28*
\_indsc: 13.
                                                         file_types: <u>10</u>,* 14, 16.
actual_input: 16, 17, 49, 50, 51, 52, 54, 55, 56, 57.
                                                         fopen: 27* 38* 40* 41*
                                                         fp: 19*, 20*, 21, 24*, 25.
argc: 3^*, 61.
argv: 3, 61, 62, 63, 71,
                                                         fprintf: 29,*30, 31, 36,*48,*53, 54, 55, 57, 59,*
banner: <u>1</u>* 60*
                                                         fputs: 68*
boolean: 4,* 19,* 22, 42,* 47,* 48.*
                                                         free: 25, 27, 40, 41.
buf_size: 6*, 14, 63*
                                                         fullname: 27,* 40,* 41.*
buffer: 14, 19,* 20,* 24,* 26, 29,* 30, 42,* 43,* 44,*
                                                         get_line: 19,* 40,* 44,* 45, 56.
    45, 46* 47* 48* 58, 63*
                                                         aetc: 20* 24*
buffer_end: 14, 20,* 24,* 63.*
                                                         history: 7,* 29,* 31, 32,* 34, 36,*
c: 20* 24*
                                                         i: 19* 29* 41* 42* 43* 47* 48* 58.
                                                         ignore: 10,* 19,* 21, 51, 58.
Cannot find change file: 41*
Cannot find master file: 40*
                                                         in_file_modes: 10.* 14.
                                                         inc_desc: 19,* 24,* 25, 30.
Cannot open change file: 41*
Cannot open include file: 27*
                                                         Include file name ...: 26, 27*
Cannot open master file: 40*
                                                         include_description: 13, 14, 19, 25, 27, 30.
                                                         init_change_file: 41,* 43,* 50.
Cannot open/create output file: 38*
ccode: 43* 44*
                                                         inp_desc: 19,* 20,* 21, 23, 24,* 25, 26, 27,* 30, 43,*
                                                              44*, 45, <u>47</u>*, <u>48</u>*, <u>50</u>, <u>63</u>*.
Change file ended without @z: 47*
Change file ended...: 45.
                                                         Input line too long: 20^*, 24^*
Change file entry ...: 58.
                                                         input_description: 14, 18, 19, 30, 42, 43, 47,
chf: 10,* 52, 55, 62, 63.*
                                                              48* 50. 63*
const_string: 4,* 66,* 67,* 68,* 69,*
                                                         input_has_ended: 21, 22, 49, 57.
                                                         input_organisation: 18, 19, 30, 40, 41, 42, 43, 46,
copyright: 1, 60.*
                                                              47* 48* 49, 50, 51, 54, 55, 56, 58, 61, 63*
CTIEHELP: 66* 68*
current_include: 14, 19, 25, 27, 30, 63,
                                                         iptr: 42*
CWEBINPUTS: 27,* 70,* 71.*
                                                         isupper: 6*
do\_includes: 19*
                                                         j: 42*, 46*
dont_match: 14, 43, 48, 51.
                                                         jptr: \underline{42}^*
e\_of\_ch\_module: 47, 50.
                                                         k: 20*, 24*, 27*, 29*
e\_of\_ch\_preamble: 48, 56.
                                                         kpathsea_version_string: 60, 69,
EOF: 20* 24*
                                                         kpse\_cweb\_format: 70.*
                                                         kpse_find_cweb: 27,* 40,* 41,* 70,*
err_print: 20,* 24,* 26, 27,* 28,* 29,* 31, 44,* 45,
    47, 48, 58.
                                                         kpse_find_file: 70*
errno: 36*
                                                         kpse_program_name: 71*
exit: 3,* 31, 36,* 59,* 68,* 69,*
                                                         kpse\_set\_program\_name: 71.*
EXIT_FAILURE: 32* 59*
                                                         l: 29*
EXIT_SUCCESS: 32, 68, 69.*
                                                         limit: 14, 19, 20, 21, 24, 26, 27, 30, 42, 43, 44,
false: 4, 19, 21, 22, 44, 47, 48, 51, 57.
                                                              45, 46, 47, 48, 63.
fatal: 7* 31, 34, 36*
                                                         line: 13, 14, 19, 23, 24, 27, 30, 63.
fatal_error: 27,* 31, 40,* 41,* 50, 63,*
                                                         lines\_dont\_match: 42,* 51.
fclose: 21, 25.
                                                         lmt: 46*
feof: 20* 24*
                                                         loc: 14, 20, 24, 26, 27, 30, 44, 48, 58, 63.
fflush: 23, 29*
                                                         main: 3^*
file_index: 12, 16, 19, 28, 29, 41, 42, 43, 46,
                                                         malloc: 9, 27, 63.
    47, 48, 49, 58.
                                                         master: 10, 21, 23, 30, 40, 50, 54, 62.
```

18

```
max_file_index: 6* 12, 18, 61.
max_file_name_length: 13, 15, 27,
max\_include\_files: 15^*.26.
message: 68.*
Missing 0x...: 44*
mode: 14, 19, 21, 50, 51, 56, 58, 63, 58
name: 69* 70*
new\_inc: 27^*
No memory for descriptor: 63*
no_ch: 16, 41, 51, 58, 61, 63,
none: 51, 52, 53, 54, 55, 56, 63*
normal: 11* 17, 53, 55.
out_file: 38* 39* 46* 53, 54, 55, 57.
out_md_type: 11,* 16.
out_mode: 16, 17, 53, 54, 55, 57.
out_name: 38* 39* 63*
parent: 13, 25, 27*
pfatal_error: 35,* 36,* 38,* 40,* 41.*
post: 11,* 54, 55, 57.
pre: 11,* 53, 54.
print\_version\_and\_exit: 65, 67* 69*
printf: 23, 34, 40, 41, 60, 69,
prod_chf: 16, 52, 61, 62.
ptr: 46*
put_line: 46,* 52, 54, 55.
putc: 29* 30, 46*
putchar: 23, 68*
puts: 69*
reading: 10,* 50, 51, 56.
restart: 19,* 25, 26, 27.*
return_code: 7.*
s: 29* 36*
search: 10,* 50, 51, 63.*
spotless: 7* 32* 34.
stderr: 8,* 29,* 30, 31, 34, 36,* 48,* 59.*
stdin: 8*
stdout: 8,* 23, 34, 68.*
strcmp: 61.
strerr: \underline{36}^*
strerror: 36*
string: 3,* 4,* 14, 27,* 39,* 40,* 41.*
strncmp: 42*
system dependencies: 6, 9, 30, 32, 34, 70.
t: 36*
tab character expansion: 19,* 20.*
temp: 25.
test: 10^*, 51.
test_file: 49, 51.
test_input: 16, 51, 52, 53, 54, 55, 56.
the_file: 13, 14, 19, 27, 40, 41.
This can't happen...: 50.
tolower: 44*
```

```
Too many nested includes: 26.
too_long: 27*
total_include_files: 15*, 25, 26, 27*
troublesome: 7,* 29,* 34.
true: 4,* 19,* 21, 40,* 42,* 45, 47,* 48,* 56, 70,*
type_of_file: 14, 21, 23, 30, 40, 50, 54, 55, 63.
ungetc: 20* 24*
unknown: 10* 16, 61, 62.
usage_error: 59* 61, 62.
usage_help: 64, 67, 68.
version: 69*
version\_number: 1,* 65.
wrap_up: 3,* 31, 32,* 33,* 36,*
xisupper: 6, 44.
```

```
(Check for Qi in newly read line, goto restart if include fails 26) Used in section 19*.
 Check that all changes have been read 58 \ Used in section 3*.
 Check the current files for any ends of changes 50 \ Used in section 49.
 Display help message and exit 64 \ Used in section 61.
 Display version information and exit 65 \ Used in section 61.
 Error handling functions 29*, 36* Used in section 2.
 Get a file name 63* Used in section 61.
 Get include line into buffer or goto restart if end of file 24* Used in section 19*.
 Get line into buffer, return false if end of file 20^* Used in section 19*.
 Get the master file started 40^* Used in section 3^*.
 Global #includes 8* Used in section 2.
 Global types 10^*, 11^*, 12, 13, 14 Used in section 2.
 Global variables 7*, 15*, 16, 18, 22, 39*, 66* Used in section 2.
 Handle end of file and return 21 \ Used in section 20*.
 Handle end of include file and goto restart 25 \ Used in section 24*.
 Handle output 52 Vsed in section 49.
 Increment the line number and print a progress report at certain times 23 \ Used in section 20*.
 Initialise parameters 17 Used in section 3^*.
 Internal functions 19^*, 32^*, 42^*, 43^*, 46^*, 47^*, 48^*, 59^* Used in section 2.
 Predeclaration of functions 28*, 33*, 35*, 67* Used in section 2.
 Prepare the change files 41^* Used in section 3^*.
 Prepare the output file 38* Used in section 3*.
 Print error location based on input buffer 30 \ Used in section 29*.
 Print the banners 60^* Used in sections 3^* and 59^*.
 Print the job history 34 \ Used in section 32^*.
 Process a line, break when end of source reached 49 \) Used in section 57.
 Process the input 57 Used in section 3^*.
 Scan all other files for changes to be done 51 \ Used in section 49.
 Scan the parameters 61 \ Used in section 3^*.
 Set a flag 62 \ Used in section 61.
 Set up PROGNAME feature and initialise the search path mechanism 71* Used in section 3*.
 Skip over comment lines; return if end of file 44* Used in section 43*.
 Skip to the next nonblank line; return if end of file 45 \ Used in section 43*.
 Step to next line 56 \ Used in section 49.
 Test for normal, break when done 53
                                            Used in section 52.
 Test for post, break when done 55 \ Used in section 52.
\langle \text{ Test for } pre, \text{ break when done } 54 \rangle Used in section 52.
 The main function 3^* Used in section 2.
(Try to open include file, abort push if unsuccessful, go to restart 27*) Used in section 26.
```