```
/* ____________________________________________________________________ *
```

src2tex* version 2.12 (Sep 9, 1996)

KAZUO AMANO† AND SHINICHI NOMOTO‡

```
* ____________________________________________________________________ */

/* ____________________________________ fileio.c ____________________________________ */

#include <stdio.h>
#include "src2tex.h"

int Page_Len_Max = -1;

/* ____________________________________ get file names in cptr[] ____________________________________ */
/* Here we get input-file-name from command line and put
        cptr[0]=input-file-name, cptr[1]=output-file-name,
Here modifying input-file-name, we determine output-file-name so that it has a file name extension of the
form *.tex .                                                                                          */

void get_fnames(argc, argv, cptr)
int argc;
char *argv[];
char *cptr[];
{
    static char workarea[2][FNAME_LEN_MAX];
    char c;
    int i;

    if ((argc != 1) && (argc != 2) && (argc != 3))
      {
#ifdef LATEX
          fprintf(stderr, "Usage: src2latex file_name\n");
#else
          fprintf(stderr, "Usage: src2tex file_name\n");
#endif
          exit(EXIT_FAILURE);
      }
    if (argc == 1)
```

* Permission to use, copy, and modify this software and its documentation is granted under no conditions. However, the authors would be very happy if users could inform any modifications to kamano@tansei.cc.u-tokyo.ac.jp. Since src2tex is a free software, there is no warranty of any kind for the program.

† Department of Mathematics, Josai University, Keyakidai, Sakado, Saitama, 350-02, JAPAN (kamano@po.iijnet.or.jp)

‡ Department of Mathematics, Josai University, Keyakidai, Sakado, Saitama, 350-02, JAPAN (m93093@crescent.josai.ac.jp)

```
      {
        workarea[0][0] = '\0';
        workarea[1][0] = '\0';
      }
    if ((argc == 2) || ((argc == 3) && (*argv[1] != '-')))
      {
        for (i = 0; i < FNAME_LEN_MAX - 1; ++i)
          {
            if ((c = *(argv[1] + i)) == '\0')
              break;
            workarea[0][i] = c;
          }
        workarea[0][i] = '\0';
        for (i = 0; i < FNAME_LEN_MAX - 5; ++i)
          {
            if ((c = *(argv[1] + i)) == '\0')
              break;
#ifndef UNIX
            if (c == '.')
              break;
#endif
            workarea[1][i] = c;
          }
        workarea[1][i] = '.';
        workarea[1][i + 1] = 't';
        workarea[1][i + 2] = 'e';
        workarea[1][i + 3] = 'x';
        workarea[1][i + 4] = '\0';
      }
    if ((argc == 3) && (*argv[1] == '-'))
      {
        Page_Len_Max = 0;
        for (i = 1; i < FNAME_LEN_MAX - 1; ++i)
          {
            if ((*(argv[1] + i) < '0') || (*(argv[1] + i) > '9'))
              break;
            Page_Len_Max = 10 * Page_Len_Max + (int)(*(argv[1] + i) - '0');
            if (Page_Len_Max >1024)
              {
                Page_Len_Max = 0;
                break;
              }
          }
#ifdef DEBUGGING
        printf ("get_fnames(): Page_Len_Max is set to %d\n", Page_Len_Max);
#endif
        for (i = 0; i < FNAME_LEN_MAX - 1; ++i)
          {
            if ((c = *(argv[2] + i)) == '\0')
              break;
            workarea[0][i] = c;
          }
        workarea[0][i] = '\0';
```

```
        for (i = 0; i < FNAME_LEN_MAX - 5; ++i)
          {
            if ((c = *(argv[2] + i)) == '\0')
              break;
#ifndef UNIX
            if (c == '.')
              break;
#endif
            workarea[1][i] = c;
          }
        workarea[1][i] = '.';
        workarea[1][i + 1] = 't';
        workarea[1][i + 2] = 'e';
        workarea[1][i + 3] = 'x';
        workarea[1][i + 4] = '\0';
      }
    cptr[0] = workarea[0];
    cptr[1] = workarea[1];
}

/* ____________________ open files for tt cptr[]=file_names, fptr[]=file_pointers ____________________ */
/* This function opens input-file and output-file with
        cptr[0]=input-file-name, cptr[1]=output-file-name
and set two file pointers
        fptr[0]=input-file-pointer, fptr[1]=output-file-pointer.                                    */

void open_files(cptr, fptr)
char *cptr[];
FILE *fptr[];
{
  if ((*cptr[0] == '-') && (*(cptr[0]+1) == 'v'))
    {
#ifdef LATEX
#ifndef UNIX
      printf("src2ltex version ");
#else
      printf("src2latex version ");
#endif
#else
      printf("src2tex version ");
#endif
      printf(VERSION);
#ifndef UNIX
      printf(" for DOS");
#else
      printf(" for UNIX");
#endif
      printf(" with %d byte buffer\n", BUFFER_SIZE);
      exit(EXIT_FAILURE);
    }
  if (*cptr[0] == '-')
    {
#ifdef LATEX
```

```
#ifndef UNIX
      printf("Usage: src2ltex [file]\n");
#else
      printf("Usage: src2latex [file]\n");
#endif
#else
      printf("Usage: src2tex [file]\n");
#endif
      printf("Here [file] should be written in one of the following languages:\n");
      printf("Basic, C, C++, Cobol, Fortran, HTML, Java, Lisp, Make,\n");
      printf("Pascal, Perl, Scheme, Shell, Tcl/Tk;\n");
      printf("Asir, Macsyma, Maple, Maxima, Mathematica, MatLab, MuPAD,\n");
      printf("Octave, Reduce.\n");
      printf("  *.tex, *.txt                =>    TEXT\n");
      printf("  *.bas, *.vb                 =>    BASIC\n");
      printf("  *.c, *.cpp, *.vc            =>    C, C++, OBJECTIVE-C\n");
      printf("  *.cbl, *.cob                =>    COBOL\n");
      printf("  *.f, *.for                  =>    FORTRAN\n");
      printf("  *.html                      =>    HTML\n");
      printf("  *.java                      =>    JAVA\n");
      printf("  *.el, *.lsp, *.sc, *.scm    =>    LISP, SCHEME\n");
      printf("  makefile                    =>    MAKE\n");
      printf("  *.p, *.pas, *.tp            =>    PASCAL\n");
      printf("  *.pl, *.prl                 =>    PERL\n");
      printf("  *.sh, *.csh, *.ksh          =>    SHELL\n");
      printf("  *.tcl, *.tk                 =>    TCL/TK\n");
      printf("  *.asi, *.asir, *.asr        =>    ASIR\n");
      printf("  *.mac, *.max                =>    MACSYMA, MAXIMA\n");
      printf("  *.map, *.mpl                =>    MAPLE\n");
      printf("  *.mat, *.mma                =>    MATHEMATICA\n");
      printf("  *.ml, *.mtlb, *.oct         =>    MATLAB, OCTAVE\n");
      printf("  *.mu                        =>    MuPAD\n");
      printf("  *.red, *.rdc                =>    REDUCE\n");
      printf("For more information, please read the src2tex.c.ps file.\n");
      exit(EXIT_FAILURE);
    }
  if (*cptr[0] == '\0')
    fptr[0] = stdin;
  else
    {
      if ((fptr[0] = fopen(cptr[0], "r")) == NULL)
        {
          fprintf(stderr, "\nError: cannot open %s\n", cptr[0]);
          exit(EXIT_FAILURE);
        }
    }
  if (*cptr[1] == '\0')
    fptr[1] = stdout;
  else
    {
      if ((fptr[1] = fopen(cptr[1], "w")) == NULL)
        {
          fprintf(stderr, "\nError: cannot open %s\n", cptr[1]);
```

```
            exit(EXIT_FAILURE);
          }
      }
}
```

/* _____ close files for cptr[]=file_names, fptr[]=file_pointers _____ */
/* This function closes two files with file pointers

fptr[0]=*input-file-pointer*, fptr[1]=*output-file-pointer*.

File names

cptr[0]=*input-file-name*, cptr[1]=*output-file-name*

are used only in error messages. */

```
void close_files(cptr, fptr)
char *cptr[];
FILE *fptr[];
{
    if (fclose(fptr[0]) == EOF)
      {
          fprintf(stderr, "\nError: cannot close %s\n", cptr[0]);
          exit(EXIT_FAILURE);
      }
    if (fclose(fptr[1]) == EOF)
      {
          fprintf(stderr, "\nError: cannot close %s\n", cptr[1]);
          exit(EXIT_FAILURE);
      }
}
```