/* ———————————————————————————————————————— *

**src2tex*** version 2.12 (Sep 9, 1996)

Kazuo Amano† and Shinichi Nomoto‡

* ———————————————————————————————————————— */

/* ———————————————— tools.c ———————————————— */

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

extern int TXT_flag;
extern int BAS_flag;
extern int C_flag;
extern int CBL_flag;
extern int F77_flag;
extern int LISP_flag;
extern int MAKE_flag;
extern int PAS_flag;
extern int PERL_flag;
extern int SH_flag;
extern int TCL_flag;

extern int MAP_flag;
extern int MAT_flag;
extern int MLAB_flag;
extern int RED_flag;

extern int *dec_buf_ptr();
extern int *inc_buf_ptr();
extern int *fgetc2buffer();
extern int *get_phrase();
extern int search_line();
extern int get_comment_flag();
extern int get_tex_flag();
extern int str_cmp();
extern int parse_options();
```

/* ———————— absorb differencies of NTT and ASCII JTEXs ———————— */



† 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)

/* Unfortunately, Japanese TEX is splitted into NTT JTEX and ASCII JTEX. Their escape sequences are often different. So, we write the following lines at the beginning of output TEX file and absorb those differences. */

```
void merge_ntt_ascii(fptr)
FILE *fptr[];
{
#ifdef ASCII
  /* NTT+ASCII JTEX                                                    */
  fprintf(fptr[1], "\\ifx\\gtfam\\undefined\n");
  fprintf(fptr[1], "  \\ifx\\dm\\undefined\n");
  fprintf(fptr[1], "    \\ifx\\tendm\\undefined\n");
  fprintf(fptr[1], "      \\def\\mc{\\null}\n");
  fprintf(fptr[1], "    \\else\n");
  fprintf(fptr[1], "      \\def\\mc{\\tendm}\n");
  fprintf(fptr[1], "    \\fi\n");
  fprintf(fptr[1], "  \\else\n");
  fprintf(fptr[1], "    \\def\\mc{\\dm}\n");
  fprintf(fptr[1], "  \\fi\n");
  fprintf(fptr[1], "  \\ifx\\dg\\undefined\n");
  fprintf(fptr[1], "    \\ifx\\tendg\\undefined\n");
  fprintf(fptr[1], "      \\def\\gt{\\null}\n");
  fprintf(fptr[1], "    \\else\n");
  fprintf(fptr[1], "      \\def\\gt{\\tendg}\n");
  fprintf(fptr[1], "    \\fi\n");
  fprintf(fptr[1], "  \\else\n");
  fprintf(fptr[1], "    \\def\\gt{\\dg}\n");
  fprintf(fptr[1], "  \\fi\n");
  fprintf(fptr[1], "\\fi\n");
  fprintf(fptr[1], "\\ifx\\sc\\undefined\n");
  fprintf(fptr[1], "  \\def\\sc{\\null}\n");
  fprintf(fptr[1], "\\fi\n");
#else
  /* NTT JTEX                                                          */
  fprintf(fptr[1], "\\ifx\\dm\\undefined\n");
  fprintf(fptr[1], "  \\def\\dm{\\tendm}\n");
  fprintf(fptr[1], "\\fi\n");
  fprintf(fptr[1], "\\ifx\\dg\\undefined\n");
  fprintf(fptr[1], "  \\def\\dg{\\tendg}\n");
  fprintf(fptr[1], "\\fi\n");
  fprintf(fptr[1], "\\ifx\\sc\\undefined\n");
  fprintf(fptr[1], "  \\def\\sc{\\null}\n");
  fprintf(fptr[1], "\\fi\n");
#endif
}
```

/* ———————————— get flagged character ———————————— */

/* This function substitutes

$$\texttt{ptr->flag} = \begin{cases} 0 & \text{when text mode} \\ 1 & \text{when quasi-}\TeX\text{ mdoe} \\ 2 & \text{when }\TeX\text{ mode ,} \end{cases}$$

$$\texttt{ptr->character} = \text{a character read from a given file}$$

and

ptr− > buffer = buffer address .
After that, it returns flag_char pointer ptr .
*Remark.* In case of PASCAL, it is very important to delay getting mode flag by observing the value of flag
prev_flag .
Since we cannot translate a phrase
{\(...\)}
or something like that properly without delaying mode transitions. */

```
flag_char *get_flag_char(fptr)
FILE *fptr[];
{
    static flag_char ptr[1];            /* flagged character      */
    static int prev_flag = 0;           /* previous flag          */
    int flag;
    int *buf_ptr;

    buf_ptr = fgetc2buffer(fptr);
    flag = get_comment_flag(buf_ptr);
    if ((prev_flag != 0) && (flag != 0) && (TXT_flag == 0))
      if (get_tex_flag(buf_ptr) != 0)
        ++flag;
    ptr->flag = flag;
    ptr->character = *buf_ptr;
    ptr->buffer = buf_ptr;
    prev_flag = flag;
    return ptr;
}
```

/* ———————— fprintf documentstyle ———————— */
/* This function simply outputs a line of the form
\documentstyle[ LATEXOPTION ]{ LATEXSTYLE }
where LATEXOPTION and LATEXSTYLE are defined in *src2tex.h*. If there exists a string "{\documentstyle....}" at the beginning of input file, then **src2latex** moves it to the top of output file. */

```
void fprintf_documentstyle(buf_ptr,fptr)
int *buf_ptr;
FILE *fptr[];
{
  int i, *b_ptr, *tail_ptr;
  char c1, c2, mini_buffer[256];

  b_ptr = get_phrase(buf_ptr,"{\\documentstyle");
  tail_ptr = b_ptr;
  while ((*tail_ptr != '}') && (*tail_ptr >= ' '))
    ++tail_ptr;
  c1 = *(tail_ptr - 1);
  c2 = *tail_ptr;
  if (((char)*b_ptr == 0x00)
      || (c1 < '0') || ((c1 > '9') && (c1 < 'A'))
      || ((c1 > 'Z') && (c1 < 'a')) || (c1 > 'z')
      || (c2 != '}'))
    {
```

```
#ifdef LATEXSTYLE
      fprintf(fptr[1],"\\documentstyle");
#ifdef LATEXOPTION
      fprintf(fptr[1],"[");
      fprintf(fptr[1],LATEXOPTION);       /* LaTeX option                                  */
      fprintf(fptr[1],"]");
#endif
      fprintf(fptr[1],"{");
      fprintf(fptr[1],LATEXSTYLE);        /* LaTeX style                                   */
      fprintf(fptr[1],"}");
      fprintf(fptr[1],"\n");
#endif
#ifdef DEBUGGING
      printf("outputting the default \"\\documentstyle ...\"\n");
#endif
    }
  else
    {
      for (i=0; ((i < 255) && ((char)*b_ptr >= ' ')); ++i)
        {
          mini_buffer[i] = (char)*++b_ptr;
          /* Here we replace the original string \documentstyle ... with a string \null ... . */
          switch(i)
            {
            case 0:
              *b_ptr = '\\';
              break;
            case 1:
              *b_ptr = 'n';
              break;
            case 2:
              *b_ptr = 'u';
              break;
            case 3:
              *b_ptr = 'l';
              break;
            case 4:
              *b_ptr = 'l';
              break;
            default:
              *b_ptr = 0x20;
            }
          if (mini_buffer[i] == '}')
            {
              ++i;
              break;
            }
        }
      mini_buffer[i] = 0x00;
      fprintf(fptr[1],"%s\n", mini_buffer);
#ifdef DEBUGGING
      printf("outputting a string \"%s...\"\n", mini_buffer);
#endif
```

```
    }
  fprintf(fptr[1],"\\begin{document}\n");
  fprintf(fptr[1],"\n");
  /* We define sevenrm escape sequence here, since it is not defined in some LaTeX systems.    */
  fprintf(fptr[1], "\\ifx\\sevenrm\\undefined\n");
  fprintf(fptr[1], "  \\font\\sevenrm=cmr7 scaled \\magstep0\n");
  fprintf(fptr[1], "\\fi\n");
}


/* ---------------------------------- fprintf footline ---------------------------------- */
/* This function simply outputs a line of the form
    \footline={\rm\hfill file-name  \qquad\folio}
where file-name is a string stored at cptr[0].                                                */

void fprintf_footline(cptr, fptr)
char *cptr[];
FILE *fptr[];
{
    char *ptr;

    fprintf(fptr[1], "\\footline={\\rm\\hfill ");
    for (ptr = cptr[0]; *ptr != '\0'; ++ptr)
      {
        switch (*ptr)
          {
          case '"':
              fprintf(fptr[1], "{\\tt \"}");
              break;
          case '#':
              fprintf(fptr[1], "{\\tt\\#}");
              break;
          case '$':
              fprintf(fptr[1], "{\\tt\\$}");
              break;
          case '%':
              fprintf(fptr[1], "{\\tt\\%c}", *ptr);
              break;
          case '&':
              fprintf(fptr[1], "{\\tt\\&}");
              break;
          case '*':
              fprintf(fptr[1], "{\\tt *}");
              break;
          case '-':
              fprintf(fptr[1], "{\\tt -}");
              break;
          case '/':
              fprintf(fptr[1], "{\\tt /}");
              break;
          case '<':
              fprintf(fptr[1], "{\\tt <}");
              break;
```

```
            case '>':
                fprintf(fptr[1], "{\\tt >}");
                break;
            case '\\':
                fprintf(fptr[1], "$\\backslash$");
                break;
            case '^':
                fprintf(fptr[1], "$\\hat{\\ }$");
                break;
            case '_':
                fprintf(fptr[1], "{\\tt\\_}");
                break;
            case '{':
                fprintf(fptr[1], "$\\{$");
                break;
            case '|':
                fprintf(fptr[1], "{\\tt |}");
                break;
            case '}':
                fprintf(fptr[1], "$\\}$");
                break;
            case '~':
                fprintf(fptr[1], "$\\tilde{\\ }$");
                break;
            default:
                fprintf(fptr[1], "%c", *ptr);
            }
      }
    fprintf(fptr[1], "\\qquad page \\folio}\n");
}
```

/* ———————————— input user's style file ———————————— */
/* This function `input_usr_style()` simply tries to input either *src2tex.s2t* or *src2latex.s2t* at the beginning of output operation. If you want to customize src2tex [resp. src2latex], it will suffice to write a style file *src2tex.s2t* [resp. *src2latex.s2t*]. */

```
void input_user_style(fptr)
FILE *fptr[];
{
#ifdef LATEX
  fprintf(fptr[1], "\\newread\\MyStyle\n");
#ifndef UNIX
  fprintf(fptr[1], "\\openin\\MyStyle=src2ltex.s2t\n");
#else
  fprintf(fptr[1], "\\openin\\MyStyle=src2latex.s2t\n");
#endif
  fprintf(fptr[1], "\\ifeof\\MyStyle\n");
  fprintf(fptr[1], "  \\closein\\MyStyle\n");
  fprintf(fptr[1], "\\else\n");
#ifndef UNIX
  fprintf(fptr[1], "  \\input src2ltex.s2t\n");
#else
```

```
  fprintf(fptr[1], "  \\input src2latex.s2t\n");
#endif
  fprintf(fptr[1], "  \\closein\\MyStyle\n");
  fprintf(fptr[1], "\\fi\n");
#ifdef DEBUGGING
  printf ("input_user_style(): src2latex.s2t is included\n");
#endif
#else
  fprintf(fptr[1], "\\newread\\MyStyle\n");
  fprintf(fptr[1], "\\openin\\MyStyle=src2tex.s2t\n");
  fprintf(fptr[1], "\\ifeof\\MyStyle\n");
  fprintf(fptr[1], "  \\closein\\MyStyle\n");
  fprintf(fptr[1], "\\else\n");
  fprintf(fptr[1], "  \\input src2tex.s2t\n");
  fprintf(fptr[1], "  \\closein\\MyStyle\n");
  fprintf(fptr[1], "\\fi\n");
#ifdef DEBUGGING
  printf ("input_user_style(): src2tex.s2t is included\n");
#endif
#endif
}
```

/* ———————————————— set TT_flag of text2tex() ———————————————— */

/* This function `choose_tt_font()` returns 1 if and only if it is better to choose typewriter font in quasi-TEX mode. Actually, this function tries to determine whether or not the user prefers cmtt font to cmr font. */

```
int choose_tt_font(buf_ptr)
int *buf_ptr;
{
  int *b_ptr, char_counter, tt_flag, tex_flag;
  int line_length0, line_length1, line_length2;

  char_counter = 0;
  line_length0 = 0;
  line_length1 = 0;
  line_length2 = 0;
  tt_flag = 0;
  tex_flag = 0;
  for (b_ptr = buf_ptr; b_ptr != buf_ptr + (int)(BUFFER_SIZE / 2); ++b_ptr)
    {
      if (*b_ptr == '\t')
        char_counter += (int)(HTAB_SKIP) - (char_counter % (int)(HTAB_SKIP));
      else
        ++char_counter;
      if (((*b_ptr == '{') && (*(b_ptr + 1) == '\\')) || (*b_ptr == '$'))
        ++tex_flag;
#ifdef UNIX
      if (*b_ptr != '\n')
        continue;
#else
      if (*b_ptr != '\r')
        continue;
```

```
#endif
      else
        {
          line_length0 = line_length1;
          line_length1 = line_length2;
          if (tex_flag == 0)
            line_length2 = char_counter;
          else
            line_length2 = 0;
          char_counter = 0;
          tex_flag = 0;
        }
      if ((line_length0 == 0) || (line_length1 == 0) || (line_length2 == 0)
          || (line_length0 != line_length1) || (line_length1 != line_length2))
        continue;
      if (C_flag != 0)
        {
          if ((*(b_ptr - 2) == '*') && (*(b_ptr -1) == '/'))
            ++tt_flag;
          if ((*(b_ptr - 2) == '/') && (*(b_ptr -1) == '/'))
            ++tt_flag;
        }
      if (CBL_flag != 0)
        {
          if (*(b_ptr - 1) == '*')
            ++tt_flag;
        }
      if (F77_flag != 0)
        {
          if (*(b_ptr - 1) == '*')
            ++tt_flag;
        }
      if (PAS_flag != 0)
        {
          if ((*(b_ptr - 2) == '*') && (*(b_ptr -1) == ')'))
            ++tt_flag;
          if (*(b_ptr - 1) == '}')
            ++tt_flag;
        }
      if (MAP_flag != 0)
        {
          if (*(b_ptr - 1) == '#')
            ++tt_flag;
        }
      if (MAT_flag != 0)
        {
          if ((*(b_ptr - 2) == '*') && (*(b_ptr -1) == ')'))
            ++tt_flag;
        }
      if (MLAB_flag != 0)
        {
          if (*(b_ptr - 1) == '#')
            ++tt_flag;
```

```
          if (*(b_ptr - 1) == '%')
            ++tt_flag;
        }
      if (tt_flag != 0)
        break;
    }
#ifdef DEBUGGING
  printf ("choose_tt_font():\n");
  printf ("TT_flag =%d\n", tt_flag);
#endif
  return tt_flag;
}
```

/* ——————————————— fprintf line numer ——————————————— */
/* This function `fprintf_line_numer()` simply prints line number at the beginning of each line. */

```
void fprintf_line_number(fptr, line_counter)
FILE *fptr[];
long line_counter;
{
  if (line_counter < 9)
    fprintf(fptr[1],
            "\n\n{\\tt\\noindent\\phantom{00000}%d:\\ }\n",
            line_counter + 1);
  if ((line_counter >= 9) && (line_counter < 99))
    fprintf(fptr[1],
            "\n\n{\\tt\\noindent\\phantom{0000}%d:\\ }\n",
            line_counter + 1);
  if ((line_counter >= 99) && (line_counter < 999))
    fprintf(fptr[1],
            "\n\n{\\tt\\noindent\\phantom{000}%d:\\ }\n",
            line_counter + 1);
  if ((line_counter >= 999) && (line_counter < 9999))
    fprintf(fptr[1],
            "\n\n{\\tt\\noindent\\phantom{00}%d:\\ }\n",
            line_counter + 1);
  if ((line_counter >= 9999) && (line_counter < 99999))
    fprintf(fptr[1],
            "\n\n{\\tt\\noindent\\phantom{0}%d:\\ }\n",
            line_counter + 1);
  if (line_counter >= 99999)
    fprintf(fptr[1],
            "\n\n{\\tt %d:\\ }\n",
            line_counter + 1);
}
```