/* ———————————————————————————————— *

src2tex* version 2.12 (Sep 9, 1996)

Kazuo Amano† and Shinichi Nomoto‡

* ———————————————————————————————— */

/* —————————————— options.c —————————————— */

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

extern int *inc_buf_ptr();
extern int *dec_buf_ptr();

int Htab_Size = HTAB_SKIP;              /* horizontal tabulation skip amount      */
int Vtab_Size = VTAB_SKIP;              /* vertical tabulation skip amount        */
#ifdef ASCII
/* ASCII JTeX                                                                     */
char Bold[] = "\\bf\\gt ";              /* bold type                              */
char Italic[] = "\\it\\mc ";            /* italic type                            */
char Roman[] = "\\rm\\mc ";             /* roman type                             */
char SmallCaps[] = "\\sc\\gt ";         /* small caps type                        */
char Slant[] = "\\sl\\mc ";             /* slanted type                           */
char Typewriter[] = "\\tt\\mc ";        /* typewriter type                        */
#else
#ifdef NTT
/* NTT JTeX                                                                       */
char Bold[] = "\\bf\\dg ";              /* bold type                              */
char Italic[] = "\\it\\dm ";            /* italic type                            */
char Roman[] = "\\rm\\dm ";             /* roman type                             */
char SmallCaps[] = "\\sc\\dg ";         /* small caps type                        */
char Slant[] = "\\sl\\dm ";             /* slanted type                           */
char Typewriter[] = "\\tt\\dm ";        /* typewriter type                        */
#else
/* TeX                                                                            */
char Bold[] = "\\bf ";                  /* bold type                              */
char Italic[] = "\\it ";                /* italic type                            */
char Roman[] = "\\rm ";                 /* roman type                             */
char SmallCaps[] = "\\sc ";             /* small caps type                        */
char Slant[] = "\\sl ";                 /* slanted type                           */
```


† 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)

```
char Typewriter[] = "\\tt ";            /* typewriter type                                  */
#endif
#endif
char *TextModeFont = Typewriter;        /* text mode font                                   */
char *TeXModeFont = Roman;              /* quasi-TEX and TEX mode font                      */
```

/* ———————— string comparison ———————— */
/* Like `strcmp()`, this function `str_cmp()` compares two strings stored at integer pointer `iptr` and character pointer `cptr`. `Str_cmp()` returns 0 if and only if given two strings are identical. */

```
int str_cmp(iptr, cptr)
int  *iptr;
char *cptr;
{
  while ((char)*iptr == *cptr)
    {
      if (*(cptr + 1) == '\0')
        break;
      iptr = inc_buf_ptr(iptr);
      ++cptr;
    }
  return (char)*iptr - *cptr;
}
```

/* ———————— ascii to integer ———————— */
/* `Asc2int()` is a substitute of `atoi()`. We just want to keep our source code closed under *stdio.h* in order to remain it highly portable. */

```
int asc2int(iptr)
int *iptr;
{
  int value = 0;

  while (((char)*iptr >= '0') && ((char)*iptr <= '9'))
    {
      value = 10 * value + (int)((char)*iptr - '0');
      iptr = inc_buf_ptr(iptr);
    }
  return value;
}
```

/* ———————— search next word ———————— */
/* This function `next_word(iptr1,iptr2)` simply increases buffer pointer `iptr1` until either it comes to the first character of the next word or it coincides with iptr2, and returns `iptr1`. */

```
int *next_word(iptr1, iptr2)
int *iptr1, *iptr2;
{
  if (iptr1 == iptr2)
    return iptr1;
```

```
  while ((iptr1 = inc_buf_ptr(iptr1)) != iptr2)
    {
      if (((((char)*(iptr1 - 1) < 'a') || ((char)*(iptr1 - 1) > 'z'))
           && (((char)*(iptr1 - 1) < '0') || ((char)*(iptr1 - 1) > '9')))
           && ((((char)*iptr1 >= 'a') && ((char)*iptr1 <= 'z'))
              || (((char)*iptr1 >= '0') && ((char)*iptr1 <='9'))))
        break;
    }
  return iptr1;
}
```

/* ———————————— parse options ———————————— */

/* `Parse_options()` changes several environment variables of **src2tex** and **src2latex** dynamically. Since using src2tex escape sequence is pretty tricky, we have decided to make very strict syntax checking before parsing options. Function `parse_options()` returns 1 , if src2tex escape sequence is found, and returns 0 otherwise. */

```
int parse_options(ptr)
flag_char *ptr;
{
  int brace_counter = 0;
  int error_flag = 0;
  int i, *iptr1, *iptr2, *iptr3;

  iptr1 = ptr->buffer;
  while (((char)*iptr1 != '\\') && ((char)*iptr1 != '\n') && (*iptr1 != EOF))
    {
      if ((((char)*iptr1 >= 'A') && ((char)*iptr1 <= 'Z'))
          || (((char)*iptr1 >= 'a') && ((char)*iptr1 <= 'z')))
        ++error_flag;
      iptr1 = inc_buf_ptr(iptr1);
    }
  /* checking syntax                                                    */
  if (str_cmp(iptr1, "\\src2tex{") != 0)
    return 0;
  if (error_flag != 0)
    {
      fprintf(stderr,
              "\nError: syntax error of src2tex escape sequence\n");
      fprintf(stderr,
              "       \\src2tex{...} is not written properly\n");
      fprintf(stderr,
              "       junks are found before \\src2tex sequence\n");
      exit(EXIT_FAILURE);
    }
  if (ptr->flag == 1)
    {
      iptr2 = dec_buf_ptr(iptr1);
      if (*iptr2 != '{')
        {
          fprintf(stderr,
                  "\nError: syntax error of src2tex escape sequence\n");
```

```
          fprintf(stderr,
                  "       \\src2tex{...} is not written properly\n");
          fprintf(stderr,
                  "       missing a left brace { \n");
          exit(EXIT_FAILURE);
        }
    }
  for (i = 0, iptr2 = iptr1; i < 256; ++i, iptr2 = inc_buf_ptr(iptr2))
    {
      if ((char)*iptr2 == '{')
        ++brace_counter;
      if ((char)*iptr2 == '}')
        --brace_counter;
      if (((char)*iptr2 == '}') && (brace_counter == 0))
        break;
      if (((char)*iptr2 == '\n') || (*iptr2 == EOF))
        break;
    }
  if (((char)*iptr2 != '}') || (brace_counter != 0))
    {
      fprintf(stderr,
              "\nError: syntax error of src2tex escape sequence\n");
      fprintf(stderr,
              "       \\src2tex{...} is not written properly\n");
      fprintf(stderr,
              "       missing right brace } \n");
      exit(EXIT_FAILURE);
    }
  iptr3 = iptr2;
  while (((char)*iptr3 != '\n') && (*iptr3 != EOF))
    {
      if ((((char)*iptr3 >= 'A') && ((char)*iptr3 <= 'Z'))
          || (((char)*iptr3 >= 'a') && ((char)*iptr3 <= 'z')))
        {
          fprintf(stderr,
                  "\nError: syntax error of src2tex escape sequence\n");
          fprintf(stderr,
                  "       \\src2tex{...} is not written properly\n");
          fprintf(stderr,
                  "       junks are found after \\src2tex sequence\n");
          exit(EXIT_FAILURE);
        }
      iptr3 = inc_buf_ptr(iptr3);
    }
#ifdef DEBUGGING
  printf("parse_options():\n");
#endif
  /* start parsing                                                      */
  for (i = 1; i == 8; ++i)
    iptr1 = inc_buf_ptr(iptr1);
  iptr1 = next_word(iptr1, iptr2);
  while(iptr1 != iptr2)
    {
```

```
      /* parsing htab option                                                */
      if (str_cmp(iptr1, "htab") == 0)
        {
          iptr1 = next_word(iptr1, iptr2);
          if (((char)*iptr1 >= '0') && ((char)*iptr1 <= '9'))
            Htab_Size = asc2int(iptr1);
        }
      /* parsing vtab option                                                */
      if (str_cmp(iptr1, "vtab") == 0)
        {
          iptr1 = next_word(iptr1, iptr2);
          if (((char)*iptr1 >= '0') && ((char)*iptr1 <= '9'))
            Vtab_Size = asc2int(iptr1);
        }
      /* TeXt font                                                          */
      if (str_cmp(iptr1, "textfont") == 0)
        {
          iptr1 = next_word(iptr1, iptr2);
          if (str_cmp(iptr1, "bf") == 0)
            TextModeFont = Bold;
          if (str_cmp(iptr1, "it") == 0)
            TextModeFont = Italic;
          if (str_cmp(iptr1, "rm") == 0)
            TextModeFont = Roman;
          if (str_cmp(iptr1, "sc") == 0)
            TextModeFont = SmallCaps;
          if (str_cmp(iptr1, "sl") == 0)
            TextModeFont = Slant;
          if (str_cmp(iptr1, "tt") == 0)
            TextModeFont = Typewriter;
        }
      /* TeX font                                                           */
      if (str_cmp(iptr1, "texfont") == 0)
        {
          iptr1 = next_word(iptr1, iptr2);
          if (str_cmp(iptr1, "bf") == 0)
            TeXModeFont = Bold;
          if (str_cmp(iptr1, "it") == 0)
            TeXModeFont = Italic;
          if (str_cmp(iptr1, "rm") == 0)
            TeXModeFont = Roman;
          if (str_cmp(iptr1, "sc") == 0)
            TeXModeFont = SmallCaps;
          if (str_cmp(iptr1, "sl") == 0)
            TeXModeFont = Slant;
          if (str_cmp(iptr1, "tt") == 0)
            TeXModeFont = Typewriter;
        }
      iptr1 = next_word(iptr1, iptr2);
    }
#ifdef DEBUGGING
  printf("Htab_Size=%d\n",Htab_Size);
  printf("Vtab_Size=%d\n",Vtab_Size);
```

```
  printf("TextModeFont=%s\n",TextModeFont);
  printf("TeXModeFont=%s\n",TeXModeFont);
#endif
  return 1;
}
```