```
/*
 *  This file is a part of Ftx2TeX, a convertor from "ftx" source files
 *  to TeX-readable files. Ftx2TeX is a part of FarsiTeX, a Persian/English
 *  typesetting system.
 *
 *  Copyright (C) 1996 Mohammad Mahdian <mahdian@mit.edu>
 *  Copyright (C) 1998-2001 Roozbeh Pournader <roozbeh@sharif.edu>
 *
 *  FarsiTeX is free software; you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation; either version 2 of the License, or
 *  (at your option) any later version.
 *
 *  FarsiTeX is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with FarsiTeX; if not, write to the Free Software
 *  Foundation, 59 Temple Place, Suite 330, Boston, MA 02111-1307,
 *  USA.
 *
 *  Any licensing or usage questions should be directed to Roozbeh
 *  Pournader <roozbeh@sharif.edu>.
 */

#include "char.h"

char e[NUM_ERAB[1]] = "";

struct chars[2] ch[256] =
{
    {0, 0, 1, 1, 0},            //
    {1, 1, 1, 1, 0},            //      1        \fathe
    {2, 2, 1, 1, 0},            //      2        \kasre
    {3, 3, 1, 1, 0},            //      3        \zamme
    {4, 4, 1, 1, 0},            //      4        \nasb
    {5, 5, 1, 1, 0},            //      5        \tashdid
    {6, 6, 1, 1, 0},            //      6        \aleph
    {7, 7, 1, 1, 0},            //      7        \hamze          //
    {8, 8, 1, 1, 0},            //      8
    {9, 9, 1, 1, 0},            //      9
    {10, 10, 1, 1, 0},          //      10
    {11, 11, 1, 1, 0},          //      11
    {12, 12, 1, 1, 0},          //      12   t gerd              //
    {13, 13, 1, 1, 0},          //      13
    {14, 14, 1, 1, 0},          //      14       "               //
    {15, 15, 1, 1, 0},          //      15
    {16, 16, 1, 1, 0},          //      16
    {17, 17, 1, 1, 0},          //      17
    {18, 18, 1, 1, 0},          //      18
    {19, 19, 1, 1, 1},          //      19 taa aakhar            //
```

Footnotes:
1: *char.h:42*
2: *char.h:44*

```
        {20, 20, 1, 1, 1},          //      20 zaa aakhar            //
        {21, 21, 1, 1, 0},          //      21       '               //
        {22, 22, 1, 1, 0},          //      22
        {23, 23, 1, 1, 0},          //      23       *               //
        {24, 24, 1, 1, 0},          //      24
        {25, 25, 1, 1, 0},          //      25       ~               //
        {26, 26, 1, 1, 0},          //      26
        {27, 27, 1, 1, 0},          //      27       +               //
        {28, 28, 1, 1, 0},          //      28
        {29, 29, 1, 1, 0},          //      29       <Space>         //
        {30, 30, 1, 1, 0},          //      30
        {31, 31, 1, 1, 0},          //      31
        {32, 32, 1, 0, 0},          //      32
        {33, 33, 1, 0, 0},          // !    33
        {34, 34, 1, 0, 0},          // "    34
        {35, 35, 1, 0, 0},          // #    35
        {36, 36, 1, 0, 0},          // $    36
        {37, 37, 1, 0, 0},          // %    37
        {38, 38, 1, 0, 0},          // &    38
        {39, 39, 1, 0, 0},          // '    39
        {40, 40, 1, 0, 0},          // (    40
        {41, 41, 1, 0, 0},          // )    41
        {42, 42, 1, 0, 0},          // *    42
        {43, 43, 1, 0, 0},          // +    43
        {44, 44, 1, 0, 0},          // ,    44
        {45, 45, 1, 0, 0},          // -    45
        {46, 46, 1, 0, 0},          // .    46
        {47, 47, 1, 0, 0},          // /    47
        {48, 48, 1, 0, 0},          // 0    48
        {49, 49, 1, 0, 0},          // 1    49
        {50, 50, 1, 0, 0},          // 2    50
        {51, 51, 1, 0, 0},          // 3    51
        {52, 52, 1, 0, 0},          // 4    52
        {53, 53, 1, 0, 0},          // 5    53
        {54, 54, 1, 0, 0},          // 6    54
        {55, 55, 1, 0, 0},          // 7    55
        {56, 56, 1, 0, 0},          // 8    56
        {57, 57, 1, 0, 0},          // 9    57
        {58, 58, 1, 0, 0},          // :    58
        {59, 59, 1, 0, 0},          // ;    59
        {60, 60, 1, 0, 0},          // <    60
        {61, 61, 1, 0, 0},          // =    61
        {62, 62, 1, 0, 0},          // >    62
        {63, 63, 1, 0, 0},          // ?    63
        {64, 64, 1, 0, 0},          // @    64
        {65, 65, 1, 0, 1},          // A    65
        {66, 66, 1, 0, 1},          // B    66
        {67, 67, 1, 0, 1},          // C    67
        {68, 68, 1, 0, 1},          // D    68
        {69, 69, 1, 0, 1},          // E    69
        {70, 70, 1, 0, 1},          // F    70
        {71, 71, 1, 0, 1},          // G    71
        {72, 72, 1, 0, 1},          // H    72
        {73, 73, 1, 0, 1},          // I    73
        {74, 74, 1, 0, 1},          // J    74
        {75, 75, 1, 0, 1},          // K    75
        {76, 76, 1, 0, 1},          // L    76
        {77, 77, 1, 0, 1},          // M    77
        {78, 78, 1, 0, 1},          // N    78
        {79, 79, 1, 0, 1},          // O    79
        {80, 80, 1, 0, 1},          // P    80
        {81, 81, 1, 0, 1},          // Q    81
        {82, 82, 1, 0, 1},          // R    82
        {83, 83, 1, 0, 1},          // S    83
        {84, 84, 1, 0, 1},          // T    84
        {85, 85, 1, 0, 1},          // U    85
        {86, 86, 1, 0, 1},          // V    86
        {87, 87, 1, 0, 1},          // W    87
        {88, 88, 1, 0, 1},          // X    88
        {89, 89, 1, 0, 1},          // Y    89
        {90, 90, 1, 0, 1},          // Z    90
        {91, 91, 1, 0, 0},          // [    91
        {92, 92, 1, 0, 0},          // \    92
        {93, 93, 1, 0, 0},          // ]    93
        {94, 94, 1, 0, 0},          // ^    94
        {95, 95, 1, 0, 0},          // _    95
        {96, 96, 1, 0, 0},          // `    96
        {97, 97, 1, 0, 1},          // a    97
        {98, 98, 1, 0, 1},          // b    98
        {99, 99, 1, 0, 1},          // c    99
        {100, 100, 1, 0, 1},        // d    100
        {101, 101, 1, 0, 1},        // e    101
        {102, 102, 1, 0, 1},        // f    102
        {103, 103, 1, 0, 1},        // g    103
        {104, 104, 1, 0, 1},        // h    104
        {105, 105, 1, 0, 1},        // i    105
        {106, 106, 1, 0, 1},        // j    106
        {107, 107, 1, 0, 1},        // k    107
        {108, 108, 1, 0, 1},        // l    108
        {109, 109, 1, 0, 1},        // m    109
        {110, 110, 1, 0, 1},        // n    110
        {111, 111, 1, 0, 1},        // o    111
        {112, 112, 1, 0, 1},        // p    112
        {113, 113, 1, 0, 1},        // q    113
        {114, 114, 1, 0, 1},        // r    114
        {115, 115, 1, 0, 1},        // s    115
        {116, 116, 1, 0, 1},        // t    116
        {117, 117, 1, 0, 1},        // u    117
        {118, 118, 1, 0, 1},        // v    118
        {119, 119, 1, 0, 1},        // w    119
        {120, 120, 1, 0, 1},        // x    120
        {121, 121, 1, 0, 1},        // y    121
        {122, 122, 1, 0, 1},        // z    122
        {123, 123, 1, 0, 0},        // {    123
        {124, 124, 1, 0, 0},        // |    124
        {125, 125, 1, 0, 0},        // }    125
        {126, 126, 0, 0, 0},        // ~    126
        {127, 127, 1, 0, 0},        //      127
        {48, 48, 1, 1, 0},          //      128 zero
        {49, 49, 1, 1, 0},          //      129 one
        {50, 50, 1, 1, 0},          //      130
        {51, 51, 1, 1, 0},          //      131
        {52, 52, 1, 1, 0},          //      132
        {53, 53, 1, 1, 0},          //      133
        {54, 54, 1, 1, 0},          //      134
        {55, 55, 1, 1, 0},          //      135
        {56, 56, 1, 1, 0},          //      136
        {57, 57, 1, 1, 0},          //      137 nine
        {44, 44, 1, 1, 0},          //      138 fcomma        ,
        {139, 139, 0, 1, 0},        //      139 -
        {63, 63, 1, 1, 0},          //      140         ?
        {141, 141, 1, 1, 1},        //      141 aa
        {142, 255, 0, 1, 1},        //     142 hamze
        {143, 143, 1, 1, 1},        //      143 hamze tanhaa
        {144, 144, 1, 1, 1},        //      144 aleph avval
        {145, 145, 1, 1, 1},        //      145 aleph aakhar
        {146, 176, 1, 1, 1},        //      146 b bozorg
        {147, 177, 0, 1, 1},        //      147 b koochek
        {148, 178, 1, 1, 1},        //      148 p bozorg
        {149, 179, 0, 1, 1},        //      149 p k
        {150, 180, 1, 1, 1},        //      150 t b
        {151, 181, 0, 1, 1},        //      151 t k
        {152, 182, 1, 1, 1},        //      152 s b
        {153, 183, 0, 1, 1},        //      153 s k
        {154, 184, 1, 1, 1},        //      154 j b
        {155, 185, 0, 1, 1},        //      155 j k
        {156, 186, 1, 1, 1},        //      156 ch b
        {157, 187, 0, 1, 1},        //      157 ch k
        {158, 188, 1, 1, 1},        //      158 h b
        {159, 189, 0, 1, 1},        //      159 h k
        {160, 190, 1, 1, 1},        //      160 kh b
        {161, 191, 0, 1, 1},        //      161 kh k
        {162, 192, 1, 1, 1},        //      162 dall
        {163, 193, 1, 1, 1},        //      163 zall
        {164, 194, 1, 1, 1},        //      164 r
        {165, 195, 1, 1, 1},        //      165 z
        {166, 196, 1, 1, 1},        //      166 zh
        {167, 197, 1, 1, 1},        //      167 sin b
        {168, 198, 0, 1, 1},        //      168 sin k
        {169, 199, 1, 1, 1},        //      169 shin b
        {170, 200, 0, 1, 1},        //      170 shin k
        {171, 201, 1, 1, 1},        //      171 sad b
        {172, 202, 0, 1, 1},        //      172 sad k
        {173, 203, 1, 1, 1},        //      173 zad b
        {174, 204, 0, 1, 1},        //      174 zad k
        {207, 206, 0, 1, 1},        //      175 taa
        {176, 176, 1, 1, 0},        //      176 \fathe
        {177, 177, 1, 1, 0},        //      177 \kasre
        {178, 178, 1, 1, 0},        //      178 \zamme
        {179, 179, 1, 1, 0},        //      179 \nasb
        {180, 180, 1, 1, 0},        //      180 \tashdid
        {35, 35, 1, 1, 0},          //      181         #
        {36, 36, 1, 1, 0},          //      182         $
        {37, 37, 1, 1, 0},          //      183         %
        {38, 38, 1, 1, 0},          //      184         &
        {39, 39, 1, 1, 0},          //      185         '
        {186, 186, 1, 1, 0},        //      186 \alef
        {187, 187, 1, 1, 0},        //      187 \hamze
        {188, 188, 1, 1, 0},        //      188 momayyez
        {40, 40, 1, 1, 0},          //      189         (
        {41, 41, 1, 1, 0},          //      190         )
        {134, 135, 1, 1, 1},        //      191 t gerd
        {34, 34, 1, 1, 0},          //      192 " basteh farsi
        {175, 205, 1, 1, 1},        //      193 taa aakhar
        {224, 208, 1, 1, 1},        //      194 zaa aakhar
        {39, 39, 1, 1, 0},          //      195 " baaz farsi
        {196, 196, 1, 1, 0},        //      196 \saken
        {45, 45, 1, 1, 0},          //      197 farsi dash (-)
        {46, 46, 1, 1, 0},          //      198         .
        {47, 47, 1, 1, 0},          //      199         /
        {42, 42, 1, 1, 0},          //      200 farsi *
        {126, 126, 1, 1, 0},        //      201 farsi ~
        {58, 58, 1, 1, 0},          //      202         :
        {59, 59, 1, 1, 0},          //      203         ;
        {62, 62, 1, 1, 0},          //      204         <    --> changed!
        {43, 43, 1, 1, 0},          //      205 farsi +
        {61, 61, 1, 1, 0},          //      206         =
        {60, 60, 1, 1, 0},          //      207         >    --> changed!
        {64, 64, 1, 1, 0},          //      208         @
        {93, 93, 1, 1, 0},          //      209         ]
        {92, 92, 1, 1, 0},          //      210         '\'
        {91, 91, 1, 1, 0},          //      211         [
        {94, 94, 1, 1, 0},          //      212         ^
        {95, 95, 1, 1, 0},          //      213         _
        {96, 96, 1, 1, 0},          //      214         `
        {125, 125, 1, 1, 0},        //      215         }
        {124, 124, 1, 1, 0},        //      216         |
        {217, 217, 1, 1, 0},        //      217
        {32, 32, 1, 1, 0},          //      218 farsi space
        {219, 219, 1, 1, 0},        //      219
        {220, 220, 1, 1, 0},        //      220      <Space>              //
        {33, 33, 1, 1, 0},          //      221         !
        {123, 123, 1, 1, 0},        //      222       {
        {223, 223, 0, 1, 0},        //      223         ~         //
        {210, 209, 0, 1, 1},        //      224 zaa
        {225, 225, 1, 1, 1},        //      225 ein tanhaa
        {226, 226, 1, 1, 1},        //      226 ein aakhar
        {227, 227, 0, 1, 1},        //      227 ein vasat
        {228, 228, 0, 1, 1},        //      228 ein avval
        {229, 229, 1, 1, 1},        //      229 ghein tanhaa
        {230, 230, 1, 1, 1},        //      230 ghein aakhar
        {231, 231, 0, 1, 1},        //      231 ghein vasat
        {232, 232, 0, 1, 1},        //      232 ghein avval
        {233, 211, 1, 1, 1},        //      233 f b
        {234, 212, 0, 1, 1},        //      234 f k
        {235, 213, 1, 1, 1},        //      235 ghaaf b
        {236, 214, 0, 1, 1},        //      236 ghaaf k
        {237, 215, 1, 1, 1},        //      237 k b
        {238, 216, 0, 1, 1},        //      238 k k
        {239, 217, 1, 1, 1},        //      239 g b
        {240, 218, 0, 1, 1},        //      240 g k
        {241, 219, 1, 1, 1},        //      241 l b
        {132, 133, 1, 1, 1},        //      242 laa
        {243, 220, 0, 1, 1},        //      243 l k
        {244, 221, 1, 1, 1},        //      244 mim b
        {245, 222, 0, 1, 1},        //      245 mim k
        {246, 223, 1, 1, 1},        //      246 n b
        {247, 128, 0, 1, 1},        //      247 n k
        {248, 129, 1, 1, 1},        //      248 v
        {249, 130, 1, 1, 1},        //      249 h aakhar
        {250, 250, 0, 1, 1},        //      250 h vasat
        {251, 251, 0, 1, 1},        //      251 h avval
        {252, 252, 1, 1, 1},        //      252 y aakhar
        {253, 253, 1, 1, 1},        //      253 y tanhaa
        {254, 131, 0, 1, 1},        //      254 y vasat
        {255, 255, 1, 1, 0}         //      255
    };
```

```
 1:  /*
 2:   *  This file is a part of Ftx2TeX, a convertor from "ftx" source files
 3:   *  to TeX-readable files. Ftx2TeX is a part of FarsiTeX, a Persian/English
 4:   *  typesetting system.
 5:   *
 6:   *  Copyright (C) 1996 Mohammad Mahdian <mahdian@mit.edu>
 7:   *  Copyright (C) 1998-2001 Roozbeh Pournader <roozbeh@sharif.edu>
 8:   *
 9:   *  FarsiTeX is free software; you can redistribute it and/or modify
10:   *  it under the terms of the GNU General Public License as published by
11:   *  the Free Software Foundation; either version 2 of the License, or
12:   *  (at your option) any later version.
13:   *
14:   *  FarsiTeX is distributed in the hope that it will be useful,
15:   *  but WITHOUT ANY WARRANTY; without even the implied warranty of
16:   *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
17:   *  GNU General Public License for more details.
18:   *
19:   *  You should have received a copy of the GNU General Public License
20:   *  along with FarsiTeX; if not, write to the Free Software
21:   *  Foundation, 59 Temple Place, Suite 330, Boston, MA 02111-1307,
22:   *  USA.
23:   *
24:   *  Any licensing or usage questions should be directed to Roozbeh
25:   *  Pournader <roozbeh@sharif.edu>.
26:   */
27:
28:  #define STRETCH    ((char) 137) // stretch character
29:  #define F_BSLASH   ((char) 210) // farsi back slash
30:  #define AT_SIGN    ((char) 208) // farsi at sign
31:  #define FB_OPEN    ((char) 222) // farsi open brace
32:  #define FB_CLOSE   ((char) 215) // farsi close brace
33:  #define FK_OPEN    ((char) 211) // farsi open bracket
34:  #define FK_CLOSE   ((char) 209) // farsi close bracket
35:  #define FP_OPEN    ((char) 190) // farsi open parenthesis
36:  #define FP_CLOSE   ((char) 189) // farsi close parenthesis
37:  #define F_MID      ((char) 216) // farsi vertical bar
38:  #define F_SPC      ((char) 218) // farsi space
39:  #define F_STAR     ((char) 200) // farsi star
40:  #define F_MOM      ((char) 188) // farsi momayyez
41:
42:  #define NUM_ERAB   8
43:
44:  struct chars {
45:      char first;                 /* normal case of letter in changed alphabet */
46:      char middle;                /* middle case of letter in changed alphabet */
47:      char last;                  /* indicates if the letter is in last case   */
48:      char farsi;                 /* indicates if the letter is a farsi letter */
49:      char letter;
50:  };
51:
52:  extern struct chars[1] ch[2][256];
53:  extern char e[3][];
```

Footnotes:
1: char.h:44
2: char.c:32
3: char.c:30

```
/*
 *  This file is a part of Ftx2TeX, a convertor from "ftx" source files
 *  to TeX-readable files. Ftx2TeX is a part of FarsiTeX, a Persian/English
 *  typesetting system.
 *
 *  Copyright (C) 1996 Mohammad Mahdian <mahdian@mit.edu>
 *  Copyright (C) 1998-2001 Roozbeh Pournader <roozbeh@sharif.edu>
 *
 *  FarsiTeX is free software; you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation; either version 2 of the License, or
 *  (at your option) any later version.
 *
 *  FarsiTeX is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with FarsiTeX; if not, write to the Free Software
 *  Foundation, 59 Temple Place, Suite 330, Boston, MA 02111-1307,
 *  USA.
 *
 *  Any licensing or usage questions should be directed to Roozbeh
 *  Pournader <roozbeh@sharif.edu>.
 */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "char.h"
#include "convert.h"
#include "ftx2tex.h"

#define MAX_LEN     500         // Maximum length of a line
#define NUM_CMD1    13
#define NUM_CMD2    5

int stretch = 1;
int noconvert = 0;
long int line_no;
char *e_cmd[NUM_ERAB[1] + 1] =
{"\\fathe ",
 "\\kasre ",
 "\\zamme ",
 "\\nasb ",
 "\\tashdid ",
 "\\alef ",
 "\\hamze ",
 "\\saken ",
 "\\skasre "};

char *command1[NUM_CMD1[2]] =
```

Footnotes:
[1]: *char.h:42*
[2]: *convert.c:36*

```
{"begin",
 "è‘¥",
 "end",
 "",
 "hspace",
 "vspace",
 "include",
 "input",
 "hspace*",
 "vspace*",
 "label",
 "ref",
 "cite"};

char *command2[NUM_CMD2[1]] =
{"documentstyle",
 "documentclass",
 "",
 "usepackage",
 "î‘¤Â¢Æµ"};

char *input_command[] =
{
    "\\input",
    "\\include"
};

#define NUM_INPUT_CMD ((sizeof input_command[2])/sizeof(char *))

void fconvert(char *s)
         /*
            This function converts a string in RL mode.
          */
{
    enum {
        FIRST, MIDDLE
    } state;
    int i, j, k, l, temp, spc;
    char t[MAX_LEN[3]], last[3];
    char *cmd;
    int is_cmd1, is_cmd2;

    for (i[4] = 0; i[4] < MAX_LEN[3]; i[4]++)
        t[5][i[4]] = 0;
    if (noconvert[6])
        return;
    state[7] = FIRST[8];
    for (i[4] = j[9] = is_cmd1[10] = is_cmd2[11] = 0; s[12][i[4]];) {
        if (s[12][i[4]] == FB_OPEN[13])
            is_cmd1[10] = is_cmd1[10] == 1 ? 2 : 0;
        else if (ch[14][(unsigned char) s[12][i[4]]].farsi[15] && s[12][i[4]] != F_SPC[16])
            is_cmd1[10] = 0;
        if (s[12][i[4]] == FK_OPEN[17])
```

Footnotes:
1: *convert.c:37*
2: *convert.c:75*
3: *convert.c:35*
4: *convert.c:91*
5: *convert.c:92*
6: *convert.c:40*
7: *convert.c:90*
8: *convert.c:89*
9: *convert.c:91*
10: *convert.c:94*
11: *convert.c:94*
12: *convert.c:83*
13: *char.h:31*
14: *char.c:32*
15: *char.h:48*
16: *char.h:38*
17: *char.h:33*

```
                is_cmd2[1] = is_cmd2[1] == 1 ? 2 : 0;
            else if (s[2][i[3]] == FK_CLOSE[4])
                is_cmd2[1] = is_cmd2[1] == 2 ? 3 : 0;
            else if (s[2][i[3]] == FB_OPEN[5])
                is_cmd2[1] = is_cmd2[1] & 1 ? 4 : 0;
            else if (ch[6][(unsigned char) s[2][i[3]]].farsi[7] && s[2][i[3]] != F_SPC[8])
                is_cmd2[1] = 0;

            if (ch[6][(unsigned char) s[2][i[3]]].farsi[7]) {
                if (s[2][i[3]] == F_MOM[9]) {
                    t[10][j[11]] = '\0';
                    strcat(t[10], "\\mom ");
                    j[11] = strlen(t[10]);
                    i[3]++;
                } else if (strchr(e[12], s[2][i[3]])) {     // e'raab

                    for (k[13] = i[3]; s[2][k[13]] && strchr(e[12], s[2][k[13]]); k[13]++);
                    t[10][j[11]] = '\0';
                    strcpy(last[14], t[10] + (j[11] -= (t[10][j[11] - 1] == STRETCH[15]) + 1));
                    if (last[14][0] == ' ' && strchr(e[12], s[2][i[3]]) == e[12] + 1)
                        s[2][i[3]] = '\0';
                    for (temp[16] = k[13]--, t[10][j[11]] = '\0'; k[13] >= i[3]; k[13]--)
                        strcat(t[10], e_cmd[17][strchr(e[12], s[2][k[13]]) - e[12]]);
                    strcat(t[10], last[14]);
                    j[11] = strlen(t[10]);
                    i[3] = temp[16];
                } else if ((s[2][i[3]] == FP_OPEN[18] || s[2][i[3]] == FP_CLOSE[19]) &&
                           i[3] != 0 && s[2][i[3] - 1] == F_MID[20]) {
                    if (s[2][i[3]] == FP_OPEN[18])
                        s[2][i[3]] = FP_CLOSE[19];
                    else
                        s[2][i[3]] = FP_OPEN[18];
                    t[10][j[11]++] = ch[6][(unsigned char) s[2][i[3]++]].first[21];
                } else {
                    if (i[3] && s[2][i[3] - 1] == F_BSLASH[22]) {     // if it is a command

                        if (s[2][i[3]] == FB_OPEN[5])
                            s[2][i[3]] = FB_CLOSE[23];
                        else if (s[2][i[3]] == FB_CLOSE[23])
                            s[2][i[3]] = FB_OPEN[5];
                        else {
                            for (k[13] = i[3], cmd[24] = t[10] + j[11]; ch[6][(unsigned char) s[2][i[3]]].letter[25];) {
                                t[10][j[11]++] = (state[26] == FIRST[27]) ? ch[6][(unsigned char) s[2][i[3]]].first[21] : ch[6][(unsigned char) s[2][i[3]]].middle[28]
    ;
                                state[26] = ch[6][(unsigned char) s[2][i[3]]].last[29] ? FIRST[27] : MIDDLE[30];
                                i[3]++;
                            }
                            if (s[2][i[3]] == F_STAR[31])
                                s[2][i[3]] = '*';
                            if (s[2][i[3]] == '*')
                                t[10][j[11]++] = s[2][i[3]++];
                            if (k[13] == i[3])
                                t[10][j[11]++] = ch[6][(unsigned char) s[2][i[3]++]].first[21];
```

Footnotes:
1: *convert.c:94*
2: *convert.c:83*
3: *convert.c:91*
4: *char.h:34*
5: *char.h:31*
6: *char.c:32*
7: *char.h:48*
8: *char.h:38*
9: *char.h:40*
10: *convert.c:92*
11: *convert.c:91*
12: *char.c:30*
13: *convert.c:91*
14: *convert.c:92*
15: *char.h:28*
16: *convert.c:91*
17: *convert.c:42*
18: *char.h:35*
19: *char.h:36*
20: *char.h:37*
21: *char.h:45*
22: *char.h:29*
23: *char.h:32*
24: *convert.c:93*
25: *char.h:49*
26: *convert.c:90*
27: *convert.c:89*
28: *char.h:46*
29: *char.h:47*
30: *convert.c:89*
31: *char.h:39*

```
                        t[1][j[2]] = '\0';
                        for (l[3] = 0; l[3] < NUM_CMD1[4] && strcmp(cmd[5], command1[6][l[3]]);)
                            l[3]++;
                        is_cmd1[7] = l[3] < NUM_CMD1[4];
                        for (l[3] = 0; l[3] < NUM_CMD2[8] && strcmp(cmd[5], command2[9][l[3]]);)
                            l[3]++;
                        is_cmd2[10] = l[3] < NUM_CMD2[8];
                        continue;
                    }
                }
                t[1][j[2]++] = (state[11] == FIRST[12]) ? ch[13][(unsigned char) s[14][i[15]]].first[16] : ch[13][(unsigned char) s[14][i[15]]].middle[17];
                state[11] = ch[13][(unsigned char) s[14][i[15]]].last[18] ? FIRST[12] : MIDDLE[19];
                if (stretch[20] && !ch[13][(unsigned char) s[14][i[15]]].last[18])
                    t[1][j[2]++] = STRETCH[21];
                i[15]++;
            }
        } else {                // english

            for (k[22] = i[15], spc[23] = 1; s[14][k[22]] && !ch[13][(unsigned char) s[14][k[22]]].farsi[24]; spc[23] &= s[14][k[22]++] == ' ');
            if (i[15] && s[14][i[15] - 1] == AT_SIGN[25] && s[14][k[22]] == AT_SIGN[25]) {
                for (s[14][k[22]] = ' ', j[2]--; i[15] < k[22]; t[1][j[2]++] = s[14][i[15]++]);
                i[15]++;
            } else if (i[15] && s[14][i[15] - 1] == F_BSLASH[26]) { // if it is a command

                for (k[22] = i[15], cmd[5] = t[1] + j[2]; (s[14][i[15]] >= 'A' && s[14][i[15]] <= 'Z') || (s[14][i[15]] >= 'a' && s[14][i[15]] <= 'z'); t[1][j[2]++] = s[14][
i[15]++]);
                if (s[14][i[15]] == F_STAR[27])
                    s[14][i[15]] = '*';
                if (s[14][i[15]] == '*')
                    t[1][j[2]++] = s[14][i[15]++];
                if (k[22] == i[15])
                    t[1][j[2]++] = s[14][i[15]++];
                t[1][j[2]] = '\0';
                for (l[3] = 0; l[3] < NUM_CMD1[4] && strcmp(cmd[5], command1[6][l[3]]); l[3]++);
                is_cmd1[7] = l[3] < NUM_CMD1[4];
                for (l[3] = 0; l[3] < NUM_CMD2[8] && strcmp(cmd[5], command2[9][l[3]]); l[3]++);
                is_cmd2[10] = l[3] < NUM_CMD2[8];
            } else {
                if (spc[23] || is_cmd1[7] & 6 || is_cmd2[10] & 6)
                    for (; k[22] - i[15]; t[1][j[2]++] = s[14][i[15]++]);
                else {
                    t[1][j[2]] = '\0';
                    strcat(t[1], "\\InE{}");
                    j[2] = strlen(t[1]);
                    for (; k[22] - i[15]; t[1][j[2]++] = s[14][i[15]++]);
                    t[1][j[2]] = '\0';
                    strcat(t[1], "\\EnE{}");
                    j[2] = strlen(t[1]);
                }
                if (is_cmd1[7] & 1)
                    is_cmd1[7] = 0;
                if (is_cmd2[10] & 1)
                    is_cmd2[10] = 0;
```

Footnotes:
1: *convert.c:92*
2: *convert.c:91*
3: *convert.c:91*
4: *convert.c:36*
5: *convert.c:93*
6: *convert.c:53*
7: *convert.c:94*
8: *convert.c:37*
9: *convert.c:68*
10: *convert.c:94*
11: *convert.c:90*
12: *convert.c:89*
13: *char.c:32*
14: *convert.c:83*
15: *convert.c:91*
16: *char.h:45*
17: *char.h:46*
18: *char.h:47*
19: *convert.c:89*
20: *convert.c:39*
21: *char.h:28*
22: *convert.c:91*
23: *convert.c:91*
24: *char.h:48*
25: *char.h:30*
26: *char.h:29*
27: *char.h:39*

```
      }
      state[1] = FIRST[2];
    }
  }

  for (i[3] = k[4] = 0; i[3] < j[5]; i[3]++)
    if (t[6][i[3]] == 127) {
      s[7][k[4]] = '\0';
      strcat(s[7], "\\char127 ");
      k[4] += strlen("\\char127 ");
    } else
      s[7][k[4]++] = t[6][i[3]];
  s[7][k[4]] = 0;
}


void econvert(char *s)
  /*
    This function converts a string in LR mode.
   */
{
  enum {
    FIRST, MIDDLE
  } state;
  int i, j, k, l, tmp, spc;
  char t[MAX_LEN[8]], last[3];

  if (noconvert[9])
    return;
  for (i[10] = j[11] = 0; s[12][i[10]];) {
    if (!ch[13][(unsigned char) s[12][i[10]]].farsi[14])
      t[15][j[11]++] = s[12][i[10]++];
    else {                  // farsi

      if (s[12][i[10]] == F_MOM[16]) {
        t[15][j[11]] = '\0';
        strcat(t[15], "\\mom ");
        j[11] = strlen(t[15]);
        i[10]++;
      } else {
        for (k[17] = i[10], spc[18] = 1; s[12][k[17]] && ch[13][(unsigned char) s[12][k[17]]].farsi[14]; spc[18] &= s[12][k[17]++] == F_SPC[19]);
        if (!spc[18]) {
          t[15][j[11]] = '\0';
          strcat(t[15], "\\InF{}");
          j[11] = strlen(t[15]);
        }
        for (state[20] = FIRST[21]; s[12][i[10]] && ch[13][(unsigned char) s[12][i[10]]].farsi[14]; i[10]++) {
          if (s[12][i[10]] == F_MOM[16]) {
            t[15][j[11]] = '\0';
            strcat(t[15], "\\mom ");
            j[11] = strlen(t[15]);
            i[10]++;
          } else if (strchr(e[22], s[12][i[10]])) {    // e'raab
```

Footnotes:
1: *convert.c:90*
2: *convert.c:89*
3: *convert.c:91*
4: *convert.c:91*
5: *convert.c:91*
6: *convert.c:92*
7: *convert.c:83*
8: *convert.c:35*
9: *convert.c:40*
10: *convert.c:235*
11: *convert.c:235*
12: *convert.c:227*
13: *char.c:32*
14: *char.h:48*
15: *convert.c:236*
16: *char.h:40*
17: *convert.c:235*
18: *convert.c:235*
19: *char.h:38*
20: *convert.c:234*
21: *convert.c:233*
22: *char.c:30*

```
264:
265:             for (l[1] = i[2]; s[3][l[1]] && strchr(e[4], s[3][l[1]]); l[1]++);
266:             t[5][j[6]] = '\0';
267:             strcpy(last[7], t[5] + (j[6] -= (t[5][j[6] - 1] == STRETCH[8]) + 1));
268:             for (tmp[9] = --l[1], t[5][j[6]] = '\0'; l[1] >= i[2]; l[1]--)
269:                 strcat(t[5], e_cmd[10][strchr(e[4], s[3][l[1]]) - e[4]]);
270:             strcat(t[5], last[7]);
271:             j[6] = strlen(t[5]);
272:             i[2] = tmp[9];
273:           } else {
274:             if (t[5][j[6] - 1] == '\\' && s[3][i[2]] == FB_OPEN[11])
275:                 s[3][i[2]] = FB_CLOSE[12];
276:             if (t[5][j[6] - 1] == '\\' && s[3][i[2]] == FB_CLOSE[12])
277:                 s[3][i[2]] = FB_OPEN[11];
278:             t[5][j[6]++] = (state[13] == FIRST[14]) ? ch[15][(unsigned char) s[3][i[2]]].first[16] : ch[15][(unsigned char) s[3][i[2]]].middle[17];
279:             state[13] = ch[15][(unsigned char) s[3][i[2]]].last[18] ? FIRST[14] : MIDDLE[19];
280:             if (stretch[20] && !ch[15][(unsigned char) s[3][i[2]]].last[18])
281:                 t[5][j[6]++] = STRETCH[8];
282:           }
283:         }
284:         if (!spc[21]) {
285:           t[5][j[6]] = '\0';
286:           strcat(t[5], "\\EnF{}");
287:           j[6] = strlen(t[5]);
288:         }
289:       }
290:     }
291:   }
292:   for (i[2] = k[22] = 0; i[2] < j[6]; i[2]++)
293:     if (t[5][i[2]] == 127) {
294:       s[3][k[22]] = '\0';
295:       strcat(s[3], "\\char127 ");
296:       k[22] += strlen("\\char127 ");
297:     } else if (t[5][i[2]] == F_MOM[23]) {
298:       s[3][k[22]] = '\0';
299:       strcat(s[3], "\\mom ");
300:       k[22] += strlen("\\mom ");
301:     } else
302:       s[3][k[22]++] = t[5][i[2]];
303:   s[3][k[22]] = 0;
304: }
305:
306:
307: void detab(char *s)
308:   /*
309:     This function changes all of the tab characters
310:     in the string s to spaces.
311:    */
312: {
313:   int i, j;
314:   char temp[500];
315:   for (i[24] = 0; i[24] < strlen(s[25]);) {
316:     if (s[25][i[24]] == '\t') {
```

Footnotes:
1: *convert.c:235*
2: *convert.c:235*
3: *convert.c:227*
4: *char.c:30*
5: *convert.c:236*
6: *convert.c:235*
7: *convert.c:236*
8: *char.h:28*
9: *convert.c:235*
10: *convert.c:42*
11: *char.h:31*
12: *char.h:32*
13: *convert.c:234*
14: *convert.c:233*
15: *char.c:32*
16: *char.h:45*
17: *char.h:46*
18: *char.h:47*
19: *convert.c:233*
20: *convert.c:39*
21: *convert.c:235*
22: *convert.c:235*
23: *char.h:40*
24: *convert.c:313*
25: *convert.c:307*

```
317:             strcpy(temp[1], s[2] + i[3] + 1);   // remainder of the string
318:
319:             for (j[4] = i[3]; i[3] < 8 * (1 + j[4] / 8); s[2][i[3]++] = ' ');
320:             s[2][i[3]] = '\0';
321:             strcat(s[2], temp[1]);
322:         } else
323:             i[3]++;
324:     }
325: }
326:
327: int convert(char *infile_name, char *outfile_name)
328: {
329:     FILE *infile, *outfile;
330:     char s[MAX_LEN[5]];
331:     int i;
332:     char *p, *q, *r;
333:     char temp_name[MAX_LEN[5]];
334:
335:     if (!(infile[6] = fopen(infile_name[7], "r"))) {
336:         printf("ftx2tex: cannot open input file \"%s\"\n", infile_name[7]);
337:         return -1;
338:     }
339:     if (!(outfile[8] = fopen(outfile_name[9], "w"))) {
340:         printf("ftx2tex: cannot open output file \"%s\"\n", outfile_name[9]);
341:         return -1;
342:     }
343:     fprintf(stderr, "Converting \"%s\" to \"%s\"", infile_name[7], outfile_name[9]);
344:
345:     line_no[10] = 0;
346:     while (fgets(s[11], MAX_LEN[5], infile[6])) {
347:         line_no[10]++;
348:         detab[12](s[11]);
349:         if (!(line_no[10] % 50))
350:             fprintf(stderr, " [%ld]", line_no[10]);
351:         if (s[11][strlen(s[11]) - 1] == '\n')
352:             s[11][strlen(s[11]) - 1] = '\0';  // delete \n from end of line
353:
354:         if (s[11][0] == '>')
355:             econvert[13](s[11] + 1);
356:         else if (s[11][0] == '<')
357:             fconvert[14](s[11] + 1);
358:         else {
359:             fprintf(stderr, "\nftx2tex: file \"%s\" line %ld: every line must begin with < or >.\n", infile_name[7], line_no[10]);
360:             return -1;
361:         }
362:         if (!strcmp(s[11] + 1, "\\nostretch"))
363:             stretch[15] = 0;
364:         else if (!strcmp(s[11] + 1, "\\stretch"))
365:             stretch[15] = 1;
366:         else if (!strcmp(s[11] + 1, "\\convert"))
367:             noconvert[16] = 0;
368:         else if (!strcmp(s[11] + 1, "\\noconvert"))
369:             noconvert[16] = 1;
```

Footnotes:
1: *convert.c:314*
2: *convert.c:307*
3: *convert.c:313*
4: *convert.c:313*
5: *convert.c:35*
6: *convert.c:329*
7: *convert.c:327*
8: *convert.c:329*
9: *convert.c:327*
10: *convert.c:41*
11: *convert.c:330*
12: *convert.c:307*
13: *convert.c:227*
14: *convert.c:83*
15: *convert.c:39*
16: *convert.c:40*

```
        else {
            for (i[1] = 0; i[1] < NUM_INPUT_CMD[2]; ++i[1]) {
                p[3] = s[4] + 1;
                while ((p[3] = strstr(p[3], input_command[5][i[1]])) != NULL) {
                    r[6] = q[7] = p[3] + strlen(input_command[5][i[1]]);
                    while (*r[6] == ' ')
                        ++r[6];
                    if (*r[6] == '{')
                        ++r[6];
                    while (*r[6] == ' ')
                        ++r[6];
                    if (r[6] != q[7]) {
                        q[7] = r[6];
                        while (*r[6] != ' ' && *r[6] != '}')
                            ++r[6];
                        strncpy(temp_name[8], q[7], r[6] - q[7]);
                        temp_name[8][r[6] - q[7]] = '\0';
                        add_file[9](temp_name[8], infile_name[10]);
                    }
                    p[3] = r[6];
                }
            }
            fprintf(outfile[11], "%s\n", s[4] + 1);
        }
    }
    fprintf(stderr, " [%ld]\n", line_no[12]);
    return 0;
}
```

Footnotes:
1: *convert.c:331*
2: *convert.c:81*
3: *convert.c:332*
4: *convert.c:330*
5: *convert.c:75*
6: *convert.c:332*
7: *convert.c:332*
8: *convert.c:333*
9: *ftx2tex.c:70*
10: *convert.c:327*
11: *convert.c:329*
12: *convert.c:41*

```
int convert[1](char *, char *);
```

Footnotes:
[1]: *convert.c:327*

```
/*
 *  This file is a part of Ftx2TeX, a convertor from "ftx" source files
 *  to TeX-readable files. Ftx2TeX is a part of FarsiTeX, a Persian/English
 *  typesetting system.
 *
 *  Copyright (C) 2001 Roozbeh Pournader <roozbeh@sharif.edu>
 *
 *  FarsiTeX is free software; you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation; either version 2 of the License, or
 *  (at your option) any later version.
 *
 *  FarsiTeX is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with FarsiTeX; if not, write to the Free Software
 *  Foundation, 59 Temple Place, Suite 330, Boston, MA 02111-1307,
 *  USA.
 *
 *  Any licensing or usage questions should be directed to Roozbeh
 *  Pournader <roozbeh@sharif.edu>.
 */

#include "filefunc.h"

#include <stdio.h>
#include <sys/stat.h>

#ifdef __BORLANDC__
#include <io.h>
#else
#include <unistd.h>
#endif

int file_exists(char *filename)
{
    return (access(filename[1], 0) == 0);
}

int file_is_newer(char *first, char *second)
{
    struct stat first_stat, second_stat;

    stat(first[2], &first_stat[3]);
    stat(second[4], &second_stat[5]);
    return (first_stat[3].st_mtime > second_stat[5].st_mtime);
}
```

Footnotes:
1: *filefunc.c:38*
2: *filefunc.c:43*
3: *filefunc.c:45*
4: *filefunc.c:43*
5: *filefunc.c:45*

```
#ifdef __MSDOS__
#define PATH_SEP[1] '\\'
#else
#define PATH_SEP '/'
#endif

#define FILENAME_LEN 1024

extern int file_exists[2](char *);
extern int file_is_newer[3](char *, char *);
```

Footnotes:
1: *filefunc.h:4*
2: *filefunc.c:38*
3: *filefunc.c:43*

```c
/*
 *  This file is a part of Ftx2TeX, a convertor from "ftx" source files
 *  to TeX-readable files. Ftx2TeX is a part of FarsiTeX, a Persian/English
 *  typesetting system.
 *
 *  Copyright (C) 1996 Mohammad Mahdian <mahdian@mit.edu>
 *  Copyright (C) 1998-2001 Roozbeh Pournader <roozbeh@sharif.edu>
 *
 *  FarsiTeX is free software; you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation; either version 2 of the License, or
 *  (at your option) any later version.
 *
 *  FarsiTeX is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with FarsiTeX; if not, write to the Free Software
 *  Foundation, 59 Temple Place, Suite 330, Boston, MA 02111-1307,
 *  USA.
 *
 *  Any licensing or usage questions should be directed to Roozbeh
 *  Pournader <roozbeh@sharif.edu>.
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "convert.h"
#include "filefunc.h"

#ifdef __BORLANDC__
#define strcasecmp stricmp
#endif

#define MAX_FILES       50

/* changes all slash characters to PATH_SEP */
void change_slashes(char *filename)
{
#if (PATH_SEP[1] != '/')
    int i;

    for (i = 0; filename[2][i] != '\0'; ++i)
        if (filename[2][i] == '/')
            filename[2][i] = PATH_SEP[1];
#endif
}

/* add an extension to the file if it didn't contain one */
void add_extension(char *filename, char *ext)
```

Footnotes:
1: *filefunc.h:4*
2: *ftx2tex.c:41*

```
{
    int i;

    i[1] = strlen(filename[2]);
    while (i[1] >= 0 && filename[2][i[1]] != '.' && filename[2][i[1]] != PATH_SEP[3])
        --i[1];
    if (i[1] < 0 || filename[2][i[1]] != '.') {
        strcat(filename[2], ".");
        strcat(filename[2], ext[4]);
    }
}

char *infile_name[MAX_FILES[5]], *outfile_name[MAX_FILES[5]];
int file_num;

/* add a file to the list of the files to be converted afterwards */
void add_file(char *filename, char *basefilename)
{
    char sourcename[FILENAME_LEN[6]], realname[FILENAME_LEN[6]];
    char *p;
    int i;
    static int flag = 0;

    i[7] = strlen(filename[8]);
    while (i[7] >= 0 && filename[8][i[7]] != '.'
           && filename[8][i[7]] != '/')
        --i[7];

    /* if extension-less or extension == '.tex' */
    if (i[7] < 0 || filename[8][i[7]] != '.' || strcmp(filename[8] + i[7] + 1, "tex") == 0) {
        if ((p[9] = strrchr(basefilename[10], PATH_SEP[3])) != NULL) {
            strncpy(realname[11], basefilename[10], p[9] - basefilename[10] + 1);
            strcpy(realname[11] + (p[9] - basefilename[10] + 1), filename[8]);
        } else
            strcpy(realname[11], filename[8]);

        change_slashes[12](realname[11]);
        add_extension[13](realname[11], "tex");
        for (i[7] = 0; i[7] < file_num[14]; ++i[7])
            if (strcasecmp(realname[11], outfile_name[15][i[7]]) == 0)
                return;
        strcpy(sourcename[16], realname[11]);
        strcpy(strrchr(sourcename[16], '.') + 1, "ftx");
        if (file_exists[17](sourcename[16]) &&
        (!file_exists[17](realname[11]) || file_is_newer[18](sourcename[16], realname[11]))) {
            if (file_num[14] >= MAX_FILES[5]) {
                if (!flag[19]) {
                    fprintf(stderr, "ftx2tex: maximum number of files exceeded, won't convert any more");
                    flag[19] = 1;
                }
            } else {
                infile_name[20][file_num[14]] = strdup(sourcename[16]);
                outfile_name[15][file_num[14]] = strdup(realname[11]);
```

Footnotes:
1: *ftx2tex.c:55*
2: *ftx2tex.c:53*
3: *filefunc.h:4*
4: *ftx2tex.c:53*
5: *ftx2tex.c:38*
6: *filefunc.h:7*
7: *ftx2tex.c:74*
8: *ftx2tex.c:70*
9: *ftx2tex.c:73*
10: *ftx2tex.c:70*
11: *ftx2tex.c:72*
12: *ftx2tex.c:41*
13: *ftx2tex.c:53*
14: *ftx2tex.c:67*
15: *ftx2tex.c:66*
16: *ftx2tex.c:72*
17: *filefunc.c:38*
18: *filefunc.c:43*
19: *ftx2tex.c:75*
20: *ftx2tex.c:66*

```
                ++file_num[1];
            }
        }
    }
}

int main(int argc, char *argv[])
{
    int i;

    if (argc[2] <= 1) {
        printf("\nftx2tex usage: ftx2tex input_file [output_file]\n");
        return 1;
    }
    if (!(infile_name[3][0] = (char *) malloc(strlen(argv[4][1]) + 5))) {
        printf("ftx2tex: cannot allocate memory\n");
        return 1;
    }
    strcpy(infile_name[3][0], argv[4][1]);
    change_slashes[5](infile_name[3][0]);
    add_extension[6](infile_name[3][0], "ftx");

    if (argc[2] > 2) {
        if (!(outfile_name[7][0] = (char *) malloc(strlen(argv[4][2]) + 5))) {
            printf("ftx2tex: cannot allocate memory\n");
            return 1;
        }
        strcpy(outfile_name[7][0], argv[4][2]);
        change_slashes[5](outfile_name[7][0]);
        add_extension[6](outfile_name[7][0], "tex");
    } else {
        if (!(outfile_name[7][0] = (char *) malloc(strlen(infile_name[3][0]) + 4))) {
            printf("ftx2tex: cannot allocate memory\n");
            return 1;
        }
        strcpy(outfile_name[7][0], infile_name[3][0]);
        i[8] = strrchr(outfile_name[7][0], '.') - outfile_name[7][0];
        outfile_name[7][0][i[8] + 4] = 0;
        if (strcasecmp(outfile_name[7][0] + i[8] + 1, "tex")) {
            outfile_name[7][0][i[8]] = 0;
            strcat(outfile_name[7][0], ".tex");
        } else
            outfile_name[7][0][i[8]] = 0;
    }

    file_num[1] = 1;
    for (i[8] = 0; i[8] < file_num[1]; ++i[8])
        convert[9](infile_name[3][i[8]], outfile_name[7][i[8]]);

    return 0;
}
```

Footnotes:
1: *ftx2tex.c:67*
2: *ftx2tex.c:113*
3: *ftx2tex.c:66*
4: *ftx2tex.c:113*
5: *ftx2tex.c:41*
6: *ftx2tex.c:53*
7: *ftx2tex.c:66*
8: *ftx2tex.c:115*
9: *convert.c:327*

```
int add_file[1](char *, char *);
```

Footnotes:
1: *ftx2tex.c:70*