```
/*
  utpdf/utps
  margin-aware converter from utf-8 text to PDF/PostScript

  Copyright (c) 2021 by Akihiro SHIMIZU

  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

  http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
*/

#include "utpdf.h"
#include "drawing.h"
#include "paper.h"
#include "usage.h"
#include "args.h"
#include "io.h"

int makepdf=1;
char *prog_name;

char *path2cmd(char *p){
    char *cur=p;
    while (*cur != '¥0') {
        if (*cur == '/') p=cur+1;
        cur++;
    }
    return p;
}

//
//
int main(int argc, char** argv){
    int fileindex;

    setlocale(LC_ALL, "");
    prog_name=path2cmd(argv[0]);
    makepdf = (strncmp(prog_name, MKPDFNAME, NAMELEN)==0);

    //
    // parse arguments
    //
    getargs(argc, argv);

    //
    // Draw each file
    //
    {
// pcobj stuff (pcobj: pango_cairo_print_object)
        pcobj *obj=NULL;
int out_fd, output_notspecified=(args->outfile==NULL);

        // for every inout file, do:
for (fileindex = optind; fileindex < argc; fileindex++) {
    // input file stuff
    UFILE *in_f;
    int in_fd;
    struct stat stat_b;

    // get input filename
```

```
69      args->in_fname = argv[fileindex];
70
71      // open input file
72      if (strncmp("-", args->in_fname, 3)==0){
73  in_fd = STDIN_FILENO;
74  args->in_fname="STDIN";
75  args->current_t=1;
76      } else {
77  in_fd = openfd(args->in_fname, O_RDONLY);
78      }
79      in_f = fdopen_u(in_fd, args->in_fname);
80
81      // create output file and surface
82      if (obj == NULL) {
83  // new file
84  if (makepdf) {
85                          // pdf
86      if (output_notspecified) {
87                              static char outf_store[S_LEN];
88
89                              snprintf(outf_store, S_LEN, "%s.pdf", args->in_fname);
90                              args->outfile = outf_store;
91      }
92                          if (strncmp(args->outfile, "-", S_LEN)==0) {
93  out_fd = STDOUT_FILENO;
94      } else {
95  out_fd = openfd(args->outfile, O_CREAT|O_RDWR|O_TRUNC);
96      }
97      obj = pcobj_pdf_new
98                              ((cairo_write_func_t )write_func, (void *)&out_fd,
99                               args->pwidth, args->pheight);
100 } else {
101                         // PostScript
102     if (output_notspecified) {
103 // write to STDOUT
104 args->outfile="-";
105                             out_fd = STDOUT_FILENO;
106     } else if (strncmp(args->outfile, "-", S_LEN)==0) {
107 out_fd = STDOUT_FILENO;
108     } else {
109 out_fd = openfd(args->outfile, O_CREAT|O_WRONLY|O_TRUNC);
110     }
111                     if (args->duplex) {
112                         if (args->force_duplex){
113                             obj = pcobj_ps_new
114                                 ((cairo_write_func_t )write_ps_duplex, (void *)&out_fd
    ,
115                                  args->phys_width, args->phys_height);
116                         } else {
117                             obj = pcobj_ps_new
118                                 ((cairo_write_func_t )write_func, (void *)&out_fd,
119                                  args->phys_width, args->phys_height);
120                         }
121                         cairo_ps_surface_dsc_comment
122                             (obj->surface, "%%Requirements: duplex");
123                         cairo_ps_surface_dsc_begin_setup(obj->surface);
124                         cairo_ps_surface_dsc_comment
125                             (obj->surface,
126                              "%%IncludeFeature: *Duplex DuplexNoTumble");
127                         // set orientation
128                         cairo_ps_surface_dsc_begin_page_setup (obj->surface);
129                         if (args->portrait) {
130                             cairo_ps_surface_dsc_comment
131                                 (obj->surface, "%%PageOrientation: Portrait");
132                         } else {
133                             cairo_ps_surface_dsc_comment
134                                 (obj->surface, "%%PageOrientation: Landscape");
135                         }
```

```
136                     } else {
137                         // simplex printing
138                         obj = pcobj_ps_new
139                             ((cairo_write_func_t )write_func, (void *)&out_fd,
140                              args->pwidth, args->pheight);
141                         // set orientation
142                         cairo_ps_surface_dsc_begin_page_setup (obj->surface);
143                         if (args->portrait) {
144                             cairo_ps_surface_dsc_comment
145                                 (obj->surface, "%%PageOrientation: Portrait");
146                         } else {
147                             cairo_ps_surface_dsc_comment
148                                 (obj->surface, "%%PageOrientation: Landscape");
149                         }
150                     } // if (args->duplex) else
151 } // if (makepdf) else
152                 // cr = cairo_create(surface);
153                 // obj = pcobj_new(cr);
154     } // if (surface == NULL)
155 
156             cairo_set_source_rgb(obj->cr, C_BLACK);
157 
158             if (args->current_t){
159                 time(args->mtime);
160             } else {
161                 if (fstat(in_fd, &stat_b)<0){
162                     perror("Could not fstat: ");
163                     exit(1);
164                 }
165                 *args->mtime = stat_b.st_mtime;
166             }
167 
168             //
169             draw_file(obj, in_f, args, (fileindex == (argc-1)));
170             //
171 
172             close_u(in_f);
173 
174             if (! args->one_output){
175                 // close output
176                 pcobj_free(obj);
177                 close(out_fd);
178                 obj=NULL;
179             } else {
180                 // if (fileindex < (argc-1)){
181                 //    cairo_show_page(cr); // new page for next file.
182                 //}
183             }
184         } // for (fileindex = optind; fileindex < argc; fileindex++) {
185 
186         if (args->one_output){
187             // close output
188             pcobj_free(obj);
189             close(out_fd);
190         }
191     }
192     exit(0);
193 }
194 
195 // end of utpdf.c
196 
```