```
/*
This ugly, sparsely-commented program is the source for text2pdf version
1.0.  It should be ANSI-conforming and compile on most platforms.
The only possible problems I know of are that SEEK_SET might have to be hash
defined to 0, and you might need to replace mktemp with _mktemp.

You may distribute the source or compiled versions free of charge.  You may
not alter the source in any way other than those mentioned above without
the permission of the author, Phil Smith <pns@cs.nott.ac.uk>.

Please send any comments to the author.

Copyright (c) Phil Smith, 1996
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

char *appname = "text2pdf v1.0";
char *progname = "text2pdf";

FILE *infile, *outfile;
int pageNo=0;
int pageObs[500];
int curObj=4;  /* object number being or last written */
long locations[1000];

char font[256];
char *defaultFont="Courier";
int tab=8;
int pointSize=10;
int vertSpace=12;
int lines=0;
int cols=80;  /* max chars per output line */
int columns=1;  /* number of columns */

/* Default paper is Letter size, as in distiller */
int pageHeight=792;
int pageWidth=612;

void WriteHeader(char *title){

  struct tm *ltime;
  time_t clock;

  time(&clock);
  ltime = localtime(&clock);

  fprintf(outfile, "%%PDF-1.0\n");
  locations[1]=ftell(outfile);
  fprintf(outfile, "1 0 obj\n");
  fprintf(outfile, "<<\n");
  fprintf(outfile, "/CreationDate (%s)\n", asctime(ltime));
  fprintf(outfile, "/Producer (%s (\\251 Phil Smith, 1996))\n", appname,  progna
me);
  if (title) fprintf(outfile, "/Title (%s)\n", title);
  fprintf(outfile, ">>\n");
  fprintf(outfile, "endobj\n");
```

```
  locations[2]=ftell(outfile);
  fprintf(outfile, "2 0 obj\n");
  fprintf(outfile, "<<\n");
  fprintf(outfile, "/Type /Catalog\n");
  fprintf(outfile, "/Pages 3 0 R\n");
  fprintf(outfile, ">>\n");
  fprintf(outfile, "endobj\n");

  locations[4]=ftell(outfile);
  fprintf(outfile, "4 0 obj\n");
  fprintf(outfile, "<<\n");
  fprintf(outfile, "/Type /Font\n");
  fprintf(outfile, "/Subtype /Type1\n");
  fprintf(outfile, "/Name /F1\n");
  fprintf(outfile, "/BaseFont %s\n", font);
  fprintf(outfile, ">>\n");
  fprintf(outfile, "endobj\n");
}

long StartPage(){
  long strmPos;

  locations[++curObj]=(int)ftell(outfile);
  pageObs[++pageNo]=curObj;
  fprintf(outfile, "%d 0 obj\n", curObj);
  fprintf(outfile, "<<\n");
  fprintf(outfile, "/Type /Page\n");
  fprintf(outfile, "/Parent 3 0 R\n");
  fprintf(outfile, "/Contents %d 0 R\n", ++curObj);
  fprintf(outfile, ">>\n");
  fprintf(outfile, "endobj\n");

  locations[curObj]=ftell(outfile);
  fprintf(outfile, "%d 0 obj\n", curObj);
  fprintf(outfile, "<<\n");
  fprintf(outfile, "/Length %d 0 R\n", curObj+1);
  fprintf(outfile, ">>\n");
  fprintf(outfile, "stream\n");
  strmPos=(int)ftell(outfile);

  fprintf(outfile, "BT\n");
  fprintf(outfile, "/F1 %d Tf\n", pointSize);
  fprintf(outfile, "1 0 0 1 50 %d Tm\n", pageHeight-40);
  fprintf(outfile, "%d TL\n", vertSpace);

  return strmPos;
}

void EndPage(long streamStart){
  long streamEnd;

  fprintf(outfile, "ET\n");
  streamEnd=ftell(outfile);
  fprintf(outfile, "endstream\n");
  fprintf(outfile, "endobj\n");

  locations[++curObj]=ftell(outfile);
  fprintf(outfile, "%d 0 obj\n", curObj);
  fprintf(outfile, "%d\n", streamEnd-streamStart);
```

```
  fprintf(outfile, "endobj\n");
}

void WritePages(){
  int atEOF=0;
  long beginstream;
  int lineNo, charNo;
  int ch, column;
  int padding, i;

  while (!atEOF) {
    beginstream=StartPage();
    column=1;
    while (column++ <= columns) {
      lineNo=0;
      while (lineNo++<lines && !atEOF) {
        fprintf(outfile, "(");
        charNo=0;
        while (charNo++<cols && (ch=getc(infile))!=EOF && ch!='\n') {
          if (ch>=32 && ch<=127) {
            if (ch=='(' || ch==')' || ch=='\\') fprintf(outfile, "\\");
            fprintf(outfile, "%c", (char)ch);
          } else {
            if (ch==9) {
              padding = tab - ((charNo-1) % tab);
              for (i = 1; i <= padding; i++) fprintf(outfile, " ");
              charNo+=(padding-1);
            }
            else
              /* write \xxx form for dodgy character */
              fprintf(outfile, "\\%.3o", ch);
          }
        }
        fprintf(outfile, ") '\n");
        if (ch==EOF) atEOF=1;
        /* remove \n if last line was maximum length anyway */
        if (charNo==cols+1 && (ch=getc(infile))!='\n') ungetc(ch, infile);
      }
      if (column<=columns) {
        fprintf(outfile, "1 0 0 1 %d %d Tm\n", (pageWidth/2)+25, pageHeight-40);
      }
    }
    EndPage(beginstream);
  }
}

void WriteRest(){
  long xref;
  int i;

  locations[3]=ftell(outfile);
  fprintf(outfile, "3 0 obj\n");
  fprintf(outfile, "<<\n");
  fprintf(outfile, "/Type /Pages\n");
  fprintf(outfile, "/Count %d\n", pageNo);
  fprintf(outfile, "/MediaBox [ 0 0 %d %d ]\n", pageWidth, pageHeight);
  fprintf(outfile, "/Resources <<\n");
  fprintf(outfile, "  /Font << /F1 4 0 R >>\n");
  fprintf(outfile, "  /ProcSet [ /PDF /Text ]\n");
  fprintf(outfile, ">>\n");
```

```
  fprintf(outfile, "/Kids [ ");
  for (i=1; i<=pageNo; i++) fprintf(outfile, "%d 0 R ", pageObs[i]);
  fprintf(outfile, "]\n");
  fprintf(outfile, ">>\n");
  fprintf(outfile, "endobj\n");

  xref=ftell(outfile);
  fprintf(outfile, "xref\n");
  fprintf(outfile, "0 %d\n", curObj+1);
  fprintf(outfile, "0000000000 65535 f \r");
  for (i=1; i<=curObj; i++)
    fprintf(outfile, "%.10d 00000 n \r", locations[i]);

  fprintf(outfile, "trailer\n");
  fprintf(outfile, "<<\n");
  fprintf(outfile, "/Size %d\n", curObj+1);
  fprintf(outfile, "/Root 2 0 R\n");
  fprintf(outfile, "/Info 1 0 R\n");
  fprintf(outfile, ">>\n");

  fprintf(outfile, "startxref\n");
  fprintf(outfile, "%d\n", xref);
  fprintf(outfile, "%%%%EOF\n");
}


void ShowHelp(){

  printf("\n%s [options] [filename]\n\n", progname);
  printf("  %s makes a 7-bit clean PDF file (version 1.0) from any input file.\n
", progname);
  printf("  It reads from standard input or a named file, and writes to standard
\n");
  printf("  output. Note that it doesn't write to standard output initially, but
\n");
  printf("  creates a temporary file and copies this to stdout at the end.\n");
  printf("\n  There are various options as follows:\n\n");
  printf("  -h\t\tshow this message\n");
  printf("  -f<font>\tuse PostScript <font> (must be in standard 14, default: Co
urier)\n");
  printf("  -s<size>\tuse font at given pointsize (default %d)\n", pointSize);
  printf("  -v<dist>\tuse given line spacing (default %d points)\n", vertSpace);
  printf("  -l<lines>\tlines per page (default 60, determined automatically\n\t\
tif unspecified)\n");
  printf("  -c<chars>\tmaximum characters per line (default 80)\n");
  printf("  -t<spaces>\tspaces per tab character (default 8)\n");
  printf("  -A4\t\tuse A4 paper (default Letter)\n");
  printf("  -A3\t\tuse A3 paper (default Letter)\n");
  printf("  -x<width>\tindependent paper width in points\n");
  printf("  -y<height>\tindependent paper height in points\n");
  printf("  -2\t\tformat in 2 columns\n");
  printf("  -L\t\tlandscape mode\n");
  printf("\n  Note that where one variable is implied by two options, the second
 option\n  takes precedence for that variable. (e.g. -A4 -y500)\n");
  printf("  In landscape mode, page width and height are simply swapped over bef
ore\n  formatting, no matter how or when they were defined.\n");
  printf("\n%s (c) Phil Smith, 1996\n", appname);
}

main(int argc, char **argv){
```

```
  char ch, tabchar=' ';
  int i = 1;
  int tmp, landscape = 0;
  char ofilename[256];
  char *ifilename = NULL;

  strcpy(font, "/");
  strcat(font, defaultFont);
  infile=stdin;  /* default */

  strcpy(ofilename, "tpXXXXXX");
  mktemp(ofilename);
  if (!ofilename) {
    fprintf(stderr, "%s: couldn't create unique temporary filename\n", progname)
;
    exit(0);
  }

  while (i < argc) {
    if (*argv[i] != '-') {  /* input filename */
      ifilename = argv[i];
      if (!(infile=fopen(ifilename, "r"))) {
        fprintf(stderr, "%s: couldn't open input file '%s'\n", progname, ifilena
me);
        exit(0);
      }
    } else {
      switch (*++argv[i]) {
      case 'h':
        ShowHelp();
        exit(0);
      case 'f':
        strcpy(font, "/");
        strcat(font, ++argv[i]);
        break;
      case 's':
        pointSize=atoi(++argv[i]);
        if (pointSize < 1) pointSize = 1;
        break;
      case 'v':
        vertSpace=atoi(++argv[i]);
        if (vertSpace < 1) vertSpace = 1;
        break;
      case 'l':
        lines=atoi(++argv[i]);
        if (lines < 1) lines = 1;
        break;
      case 'c':
        cols=atoi(++argv[i]);
        if (cols < 4) cols = 4;
        break;
      case '2':
        columns = 2;
        break;
      case 't':
        tab=atoi(++argv[i]);
        if (tab < 1) tab = 1;
        break;
      case 'A':
        switch (*++argv[i]) {
```

```
        case ’3’:
          pageWidth=842;
          pageHeight=1190;
          break;
        case ’4’:
          pageWidth=595;
          pageHeight=842;
          break;
        default:
          fprintf(stderr, "%s: ignoring unknown paper size: A%s\n", progname, ar
gv[i]);
        }
        break;
      case ’x’:
        pageWidth=atoi(++argv[i]);
        if (pageWidth < 72) pageWidth = 72;
        break;
      case ’y’:
        pageHeight=atoi(++argv[i]);
        if (pageHeight < 72) pageHeight = 72;
        break;
      case ’L’:
        landscape=1;
        break;
      default:
        fprintf(stderr, "%s: ignoring invalid switch: -%s\n", progname, argv[i])
;
      }
    }
    i++;
  }
  if (!(outfile=fopen(ofilename, "w+"))) {
    fprintf(stderr, "%s: couldn’t temporary file\n", progname);
    exit(0);
  }

  if (landscape) {
    tmp=pageHeight;
    pageHeight=pageWidth;
    pageWidth=tmp;
  }

  if (lines==0) lines=(pageHeight-72)/vertSpace;
  if (lines < 1) lines = 1;
  /* happens to give 60 as default */

  WriteHeader(ifilename);
  WritePages();
  WriteRest();

  /* copy tmp output file to stdout */
  fseek(outfile,0,SEEK_SET);
  while ( (ch=getc(outfile))!=EOF ) printf("%c", (char)ch);
  remove(ofilename);

  return 0;
}
```