ClibPDF Library Reference Manual Addendum

[Manual version 1.10-Addendum; 1999-06-07]

ClibPDF™ is a library of C functions for generating PDF files directly. This is a supplement/addendum to the **ClibPDF**™ manual version 1.10, until the sections below are incorporated into the main manual.

New Functions in Version 1.10-7d

char ***cpdf_rawTextBox**(float xl, float yl, float width, float height,
float linespace, CPDFtboxAttr *tbattr, char *text);
char ***cpdf_textBox**(float xl, float yl, float width, float height,
float linespace, CPDFtboxAttr *tbattr, char *text);

These two functions finally implement one of the top requested features: columnar formatting of text. The function allows simple **text-to-pdf** capability with left-, right-, centering-, and justification. Four float arguments **(xl, yl, width, height)** specify a text box into which text is formatted. **(xl, yl)** is the bottom-left coordinate of the box and **(width, height)** are the size parameters of the box. **Linespace** (given in points, 1/72 inches) spacifies the spacing between lines (distance between baseline of one line to that of the next).

The next argument, tbattr, is a pointer to a textbox attribute structure, and has the format shown below. You may pass a **NULL** as **tbattr**, in which case defalut values are used. Note that hhis structure is likely to have additional members in the future.

```
typedef struct {
    int alignmode;          /* one of the modes above (default= TBOX_LEFT) */
    int NLmode;             /* if non-zero, NL is is a line break, if 0 reformatted (default= 0) */
    float paragraphSpacing; /* extra space between paragraphs (default= 0.0) */
} CPDFtboxAttr;
```

For passing as "**alignmode**" to cpdf_textBox() and cpdf_rawTextBox():

```
#define     TBOX_LEFT       0
#define     TBOX_CENTER     1
#define     TBOX_RIGHT      2
#define     TBOX_JUSTIFY    3
```

Alignmode should be one of the above defines (from cpdflib.h).
NLmode = 0 will reformat text into the columnar width ignoreing single newline ('\n') characters in the text. Consecutive newline chars of 2 or more will be honored, and terminates a paragraph. In this mode, some non-zero **paragraphSpace** would be useful to give some separation between paragraph blocks.
NLmode = 1 will perform simple text-to-PDF formatting honoring newline ('\n') chars literally as line breaks. In this mode, **paragraphSpacing** probably should be 0 because it simply adds to **linespace** (see above) given as a main argument.

RETURN VALUE:

The return value is **NULL** if all the text fits into the text boxed defined. If the text is too long to be contained in the textbox, a **(char *) pointer to the remainder** of the text is returned. This points to somewhere in the middle of the original text string. This return value may be used as the new text string for another textbox to continue text placement into another column or to one on the next page. (See the TextBox example in examples/textbox/textbox.c.)

BUGS/POTENTIAL PROBLEMS:

This function is not likely to work for CJK (Chinese, Japanese, and Korean) language text, which may not contain any space character.

TABs don't work. (This, we will probably add at some point.)

This function modifies the portion of the text data that have been used (have already been put into a text box). If you need to retain the original text, make a copy and use that with these functions.

I don't think we will extend these text box functions much further. We know things like underline, bold, italics, changing fonts, HTML support, RTF support, etc. would be nice, but those will require a complete rewrite of cpdfTextBox.c.

EXAMPLE:

```
char *textbuf = "Some very long text string\nfrom somewhere.....\n\n";
char *currtext;
CPDFtboxAttr tboxatr;
float linespace = 12.0;

   tboxatr.alignmode = TBOX_JUSTIFY;
   tboxatr.NLmode = 0;          /* reformat */
   tboxatr.paragraphSpacing = 12.0;
   currtext = textbuf;          /* point to head of text */
   cpdf_beginText(0);
   cpdf_setFont("Times-Roman", "MacRomanEncoding", 12.0);
   currtext = cpdf_rawTextBox(72.0, 72.0, 225.0, 648.0,
               linespace, &tboxatr, currtext);          /* left column */
   currtext = cpdf_rawTextBox(72.0, 315.0, 225.0, 648.0,
               linespace, &tboxatr, currtext);          /* right column */
   cpdf_endText();
```

char ***cpdf_version**(void)

This function return the version string defined in version.h.

char ***cpdf_platform**(void)

This function returns the platform string defined in config.h for each platform.

CPDFerrorHandler **cpdf_setErrorHandler**(CPDFerrorHandler handler)

This function installs a custom error hander of the form below. Without this call, the default handling is as shown below. You may modify the behavior to use GUI-based alert panels or you may prevent the program from exiting entirely when level < 0.

```
void myErrorHandler(int level, const char* module, const char* fmt, va_list ap)
{
   if(module != NULL)
         fprintf(stderr, "%s: ", module);
   vfprintf(stderr, fmt, ap);
   fprintf(stderr, ".\n");
   if(level < 0)
         exit(level);
}
```

Notes on Added Example Programs

A. Text Box/TEXT2PDF *[examples/textbox/textbox.c]*

This program gives a simple example for the usage of cpdf_rawTextBox(). It takes a text file and format into a 2-column justified PDF file. By setting attributes, this example program may serve as a starting point for a very capable TEXT2PDF program.

[end of doc/1999-06-07;]