Haru Free PDF Library

Version 1.2.0

Contents

CHAPTER 1

Introduction

1.1 What is Haru

Haru is a library that has the ability to generate PDF document for free.
It supports most of the standerd features of the Portable Document Format.

1.2 The features of HARU.

1. Generation of PDF file with text, graphics, images and so on.
2. Using Base14 Fonts.
3. Type1 font embededing.
4. Compression of contents stream of deflator.
5. PNG and JPEG images.
6. Using Multibyte Fonts (Japanese, Chinese, Korean).
7. Generating PDF files with outline.
8. Link annotation.
9. Using font with custom encoding.
10. Vertical writing of Japanese fonts.
11. 40 bit key length encryption.

1.3 Requirements

1.3.1 C++ Compiler

Haru is written in C++ and it uses only standard features of C++. Thus with modifying the makefile, it can be used in many platforms.

1.3.2 Tested Platforms

Haru is tested in the following environment.

- **makefile.gcc** Linux + gcc (2.95, 3.2))
- **makefile.cygwin** Cygwin + gcc (2.95, 3.2)
- **makefile.bcc** Windows + Borland C++ 5.5
- **makefile.msvc** Windows + Microsoft VC++ Compiler (5.0 or later)
- **makefile.watcom** Windows + Open Watcom Compiler 1.0

- **makefile.aixgcc** AIX4.3/5L + gcc(2.9)
- **makefile.aixcc** AIX5L + IBM C/C++ Compiler
- **makefile.beos** BeOS(5.03) + gcc(2.95)

1.4 Required Libraries

1.Zlib compression library(version 1.1.X)
To use flate-decode compression, zlib is required.
Zlib compression library is available at the following URL:
http://www.gzip.org/zlib/

2.PNG Reference Library(version 1.1.X, 1.2.X)
To use png images, libpng is required.
PNG Reference Library is available at the following URL:
http://www.libpng.org/pub/png/libpng.html

3.JPEGLIB(IJG JPEG LIBRARY)
To use jpeg images, jpeglib is required.
The IJG JPEG LIBRARY is available at the following URL:
http://www.ijg.org/

CHAPTER 2

Installation

2.1 Make Library

Make files for some platforms are prepared named makefile.XXX.
If the makefile for your platform is prepared and all required libraries are installed in your pc, simply run make command as following.

· Linux + gcc	make -f makefile.gcc
· Cygwin + gcc	make -f makefile.cygwin
· Windows + Borland C++ Compiler	make -f makefile.bcc
· Windows + Open Watcom Compiler	wmake -f makefile.watcom
· Windows + Microsoft VC++ Compiler	wmake -f makefile.msvc
· AIX(4.3/5L) + gcc	make -f makefile.aixgcc
· AIX(5L) + IBM C/C++ Compiler	make -f makefile.aixcc
· BeOS(5.0.X) + gcc	make -f makefile.beos

2.2 Installation of the Library Files

If you want to install HARU in the default directory, the command is the following.

```
make -f makefile.XXX install (Change XXX to your platform)
```

The default nstallation directory for the library file.

· Linux, AIX, Cygwin	/usr/local/lib
· Beos	/home/config/lib
· Windows	the install command is not prepared.

The default installation directory for the header file.

· Linux, AIX, Cygwin	/usr/local/include
· Beos	/home/config/include
· Windows	the install command is not prepared.

In Windows + BCC, the install command is not prepared, copy libharu.lib to your library directory manually.

If you want to install HARU to your own library directory, modify the PREFIX variable of makefile.xxx.
For example, if you want to install HARU in "/home/myhome/lib" modify makefile.xxx as the following.

```
PREFIX=/home/myhome/lib
```

2.3 Compile and Run the Example Programs

To make the sample programs, Run make command as the following.

```
make -f makefile.XXX example
```

CHAPTER 3

Make a Program with Using Haru

3.1 The flow of PDF file creation.

Creation of a PDF file with HARU is following bellow.
1. Including "libharu.h" in your program.

```
#include "libharu.h"
```

2. Create a instance of PdfDoc object.

```
PdfDoc *doc = new PdfDoc();
```

3. Call NewDoc() method of a PdfDoc object to create a new PDF document.

```
doc->NewDoc();
```

4. Add Fonts which is used in the current document to the PdfDoc object.

```
doc->AddType1Font(new PdfTimesRomanFontDef());
doc->AddType1Font(new PdfHelveticaFontDef());
```

5. Call AddPage() method of a PdfDoc object and get PdfPage object.

```
PdfPage *page = doc->AddPage();
```

6. Get PdfContents object by calling Canvas() method of PdfPage object.

```
PdfContents *canvas = page->Canvas();
```

7. Create the contents of the page by calling various methods of a PdfContentns object.

```
canvas->SetFontAndSize("Times-Roman", 24);
canvas->BeginText();
canvas->MoveTextPos(150, 400);
canvas->ShowText("Hello");
canvas->EndText();
```

8. Call WriteToFile() method of PdfDoc object and save the document to file.

```
doc->WriteToFile("Hello.pdf");
```

9. Delete the instance of PdfDoc object.

```
delete doc;
```

10. To handle exception, protect from NewDoc() to WriteToFile() using try and catch block.

```
PdfDoc *doc = new PdfDoc();
try {
    doc->NewDoc();
    .
    .
    .
    doc->WriteToFile("Hello.pdf");
} catch (exception& e) {
    fprintf(stderr, "%s\n", e.what());
}
delete doc;
```

3.2 Graphics State

PdfContents holds the parameter named Graphics State which shows the state of canvas, and the method which can be used according to a state is restricted.
GraphicsState is changed by calling a drawing method and the value can be referred to by PdfContents::GMode().
PDF_GMODE_PAGE_DESCRIPTION

Is standard drawing mode. The method belonging to General graphics state, and Special graphic state and Text positioning and a MoveTo() method, and a BeginText() method can be used.
By calling a MoveTo() method, it changes to PDF_GMODE_PATH_OBJECT. Moreover, it changes to PDF_GMODE_TEXT_OBJECT mode by calling a BeginText() method.

PDF_GMODE_PATH_OBJECT

Is the mode which describes a line.

PDF_GMODE_TEXT_OBJECT

Is the mode which draws a character. By the EndText() method, it returns to PDF_GMODE_PAGE_DESCRIPTION mode.

PDF_GMODE_CLIPPING_PATH

PDF_GMODE_SHADING•APDF_GMODE_INLINE_IMAGE•APDF_GMODE_EXTERNAL_OBJECT

There is no method which uses this state now.

3.3 Using Base 14 Fonts

In Haru, it is possible to use three kinds of fonts Base14Font, Type1 Font, and CID font. Base14Font and a Type1 Font object are combining with an Encoding object, and can be dealt in various languages.
With using CID font, the 2-byte font in the Asian area will be available.

3.3.1 Base 14 Font

In Haru, the 14 following kinds of fonts currently called standard fonts are incorporated. Since these fonts do not have the necessity of surely being able to use it with the viewer application of every PDF file, and incorporating a font file, they can also make size of a PDF file small.

CHAPTER 4

The list of Base 14 Font

·Helvetica	PdfHelveticaFontDef
·Helvetica.Bold	PdfHelveticaBoldFontDef
·Helvetica.Oblique	PdfHelveticaObliqueFontDef
·Helvetica.BoldOblique	PdfHelveticaBoldObliqueFontDef
·Times.Roman	PdfTimesRomanFontDef
·Times.Bold	PdfTimesBoldFontDef
·Times.Italic	PdfTimesItalicFontDef
·Times.BoldItalic	PdfTimesBoldItalicFontDef
·Courier	PdfCourierFontDef
·Courier.Bold	PdfCourierBoldFontDef
·Courier.Oblique	PdfCourierObliqueFontDef
·Courier.BoldOblique	PdfCourierBoldObliqueFontDef
·Symbol	PdfSymbolFontDef
·ZapfDingbats	PdfZapfDingbatsFontDef

4.0.1 Predefined Encoding

The Following encodings are predefined in Haru.

·StandardEncoding	PdfStandardEncoding
·WinAnsiEncoding	PdfWinAnsiEncoding
·MacRomanEncoding	PdfMacRomanEncoding
·SymbolFontEncoding	PdfSymbolFontEncoding
·ZapfDingbatsFontEncoding	PdfZapfDingbatsFontEncoding

SymbolFontEncoding and ZapfDingbatsFontEncoding can be used among the encodings only combining a Symbol font and a ZapfDingbats font, respectively. Th kinds of other encodings can be combined with all fonts other than a Symbol font an ZapfDingbats font.

4.0.2 The usage of Base 14 Font

4.1 Type1 Font Embeding Feature.

4.2 Using Multibyte Fonts

4.3 Using Flatedecode compression.

4.4 PNG Image.

4.5 JPEG Image.

4.6 Encryption.

CHAPTER 5

Class Reference

CHAPTER 6

Operator Summary of PdfContentns Class.

6.1 General graphics state

6.1.1 SetLineWidth

Syntax

```
void SetLineWidth(
    double linewidth);
```

Arguments

linewidth
Is the width of the line drawn by path painting operator.

6.1.2 SetLineCap

Syntax

```
void SetLineCap(
    pdf_line_cap_style linecap);
```

Arguments

linecap
Is the cap style of the line drawn by path painting operator.

pdf_linecap_style

- **PDF_BUTT_END** — Butt cap. The stroke is squared off at the endpoint of the path.(Default)
- **PDF_ROUND_END** — Round cap. The stroke is ended with a semicircular arc.
- **PDF_PROJECTING_SCUARE_END** — Projecting square cap. The stroke is squared off beyond the endpoint of the path with a distance equal to half the line width.

6.1.3 SetLineJoin

Syntax

```
void SetLineJoin(
```

```
    pdf_line_join_style linejoin);
```

Arguments

linecap
Is the join style of the line drawn by path painting operator.

pdf_line_join_style

· PDF_MITER_JOIN	Miter joins
· PDF_ROUND_JOIN	Round joins
· PDF_BEVEL_JOIN	Bevel joins

6.1.4 SetMiterLimit

Syntax

```
void SetMiterLimit(
    double miterlimit);
```

Arguments

miterlimit
Is used in order to calculate mitterlength of two line segments.

6.1.5 SetDash

Syntax

```
void SetDash(
    unsigned int on,
    unsigned int off,
    unsigned int phase);
```

Arguments

on
Is the length of dashs. The default value is 0.

off
Is the length of gaps. The default value is 0.

phase
Is a number to specify at which to begin marking the path. the default value is 0.

6.1.6 SetFlat

Syntax

```
void SetFlat(
    unsigned int flatness);
```

Arguments

flatness

Is the flatness parameter in the graphics state. the parameter has to set between 0 to 100. The default is 0.

6.2 Special graphics state

6.2.1 GSave

Save the current graphics state to the stack. the graphics state restore by GRestore() method.

Syntax

```
void GSave();
```

6.2.2 GReatore

Restore the most recent graphics state saved by GSave() method.

Syntax

```
void GSave();
```

6.2.3 Concat

Set the current transformation matrix to specified values. The default matrix is [1, 0, 0, 1, 0, 0].

Syntax

```
void Concat(
    double a,
    double b,
    double c,
    double d,
    double e,
    double f);
```

6.3 Path construction

6.3.1 MoveTo

Moves the current point to the specified positioni.

Syntax

```
void MoveTo(
    double x,
    double y);
```

6.3.2 LineTo

Appends a line segment from the current point to the specified position.

Syntax

```
void LineTo(
    double x,
 double y);
```

6.3.3 CurveTo

Appends a Bézier curve from the current point to (x3, y3).
CurveTo uses (x1, y1) and (x2, y2) as the control point of a Bézier curve.

Syntax

```
void CurveTo(
    double x1,
    double y1,
    double x2,
    double y2,
    double x3,
    double y3);
```

6.3.4 CurveTo2

Appends a Bézier curve from the current point to (x3, y3).
CurveTo uses current point and (x2, y2) as the control point of a Bézier curve.

Syntax

```
void CurveTo(
    double x2,
    double y2,
    double x3,
    double y3);
```

6.3.5 CurveTo2

Appends a Bézier curve from the current point to (x3, y3).
CurveTo uses (x1, y1) and (x3, y3) as the control point of a Bézier curve.

Syntax

```
void CurveTo(
    double x1,
    double y1,
    double x3,
    double y3);
```

6.3.6 ClosePath

Appends a strait line from the current point and the starting.

Syntax

```
void ClosePath();
```

6.3.7 Rectangle

Adds a rectangle to the current path.

Syntax

```
void Rectangle(double x, double y, double width,
    double height);
```

6.4 Path painting

6.4.1 Stroke

Stroke the current path.

Syntax

```
void Stroke();
```

6.4.2 ClosePathStroke

Close the path then stroke.

Syntax

```
void ClosePathStroke();
```

6.4.3 Fill

Fill the region enclosed by the path.

Syntax

```
void Fill();
```

6.4.4 EoFill

Fill the region enclosed by the path with even-odd rule.

Syntax

```
void EoFill();
```

6.4.5 FillStroke

Fill the region enclosed by the path and stroke the path.

Syntax

```
void FillStroke();
```

6.4.6 EoFillStroke

Fill the region enclosed by the path with even-odd rule and stroke the path.

Syntax

```
void EoFillStroke();
```

6.4.7 ClosePathFillStroke

Close the path then fill the region enclosed by the path and stroke the path.

Syntax

```
void ClosePathFillStroke();
```

6.4.8 ClosePathEoFillStroke

Close the path then fill the region enclosed by the path with even-odd rule and stroke the path.

Syntax

```
void ClosePathEoFillStroke();
```

6.5 Clipping paths

6.5.1 Clip

Clipping the region enclosed by the path.

Syntax

```
void Clip();
```

6.5.2 EoClip

Clipping the region enclosed by the path with even-odd rule.

Syntax

```
void EoClip();
```

6.6 Text object

6.6.1 BeginText

Begin text object. As side effect, the Graphics State is set to PDF_GMODE_TEXT_OBJECT.

Syntax

```
void BeginText();
```

6.6.2 EndText

End text object. The Graphics State is set to PDF_GMODE_PAGE_DESCRIPTION as side effect.

Syntax

```
void EndText();
```

6.7 Text state

6.7.1 SetCharSpace

Set the character space of the text to the specified value.

Syntax

```
void SetCharSpace(
    double value);
```

6.7.2 SetWordSpace

Set the word space of the text to the specified value.

Syntax

```
void SetWordSpace(
    double value);
```

6.7.3 SetHorizontalScalling

Set the horizontal scalling of the font to the specified value.

Syntax

```
void SetHorizontalScalling(
    double value);
```

6.7.4 SetTextLeading

Set the text leading to the specified value. The default value is 0.

Syntax

```
void SetTextLeading(
    double value);
```

6.7.5 SetFontAndSize

Set the font of the text. It throws PDF_RUNTIME_ERROR when a invalid font is

specified.

Syntax

```
void SetFontAndSize(
    const char* fontname,
    double size);
```

```
void SetFontAndSize(
    PdfFont* font,
    double size);
```

Arguments

fontname
Is the name of the font registered by PdfDoc::AddType1Font.

font
Is the pointer of a PdfDoc object.

size
Is the size of the font. The size must be more than 0.

6.7.6 SetTextRenderingMode

Set the rendering mode of the character.

Syntax

```
void SetTextRenderingMode(
    pdf_text_rendering_mode mode);
```

Arguments

mode
Is the rendering mode. It must be one of the following.

pdf_text_rendering_mode

· **PDF_FILL**	Fill text by the current filling color.
· **PDF_STROKE**	Stroke text by the current stroke color.
· **PDF_FILL_THEN_STROKE**	Fill, then stroke text.
· **PDF_FILL_CLIPPING**	Fill text and add to path for clipping.
· **PDF_STROKE_CLIPPING**	Stroke text and add to path for clipping.
· **PDF_FILL_STROKE_CLIPPING**	Fill, then stroke, text and add to path for clipping.
· **PDF_CLIPPING**	Add text to path for clipping.

6.7.7 SetTextRaise

Move baseline of the text from the current position.

Syntax

```
void SetTextRaise(
```

```
    double value);
```

6.8 Text positioning

6.8.1 MoveTextPos

Move the current text position to the specified X and Y values.

Syntax

```
void MoveTextPos(
    double value);
```

6.8.2 MoveTextPos2

Move the current text position to the specified X and Y values. And set the text reading to -Y. This method is a same as the following.

```
    MoveTextPos(x, y);
    SetTextLeading(-y);
```

Syntax

```
void MoveTextPos2(
    double value);
```

6.8.3 SetTextMatrix

Syntax

```
void SetTextMatrix(
    double a,
    double b,
    double c,
    double d,
    double x,
    double y);
```

6.8.4 MoveToNextLine

Move the current text position to the next line.

Syntax

```
void MoveToNextLine();
```

6.9 Text showing

6.9.1 ShowText

Show a text at the current text position.

Syntax

```
void ShowText(
    const char* text);
```

6.9.2 ShowTextNextLine

Move to the next line and show a text.

Syntax

```
void ShowTextNextLine(
    const char* text);
void ShowTextNextLine(
    double aw,
    double ac,
    const char* text);
```

6.10 Color showing

6.10.1 SetGrayFill

Set Gray-scale color for filling operations. The gray parameter must be decimal between 0.0 to 1.0

Syntax

```
void SetGrayFill(
    double gray);
```

6.10.2 SetGrayStroke

Set Gray-scale color for stroke operations. The gray parameter must be decimal between 0.0 to 1.0

Syntax

```
void SetGrayStroke(
    double gray);
```

6.10.3 SetRGBFill

Set RGB color for filling operations.
On first format, each color is specified for the integers from 0 to 255.
On second format, each color is specified for the decimals from 0.0 to 1.0.

Syntax

```
void SetRGBFill(
    int r,
    int g,
```

```
    int b);
void SetRGBFill(
    double r,
    double g,
    double b);
```

6.10.4 SetRGBStroke

Set RGB color for stroking operations.
On first format, each color is specified for the integers from 0 to 255.
On second format, each color is specified for the decimals from 0.0 to 1.0.

Syntax

```
void SetRGBStroke(
    int r,
    int g,
    int b);
void SetRGBStroke(
    double r,
    double g,
    double b);
```

6.10.5 SetCMYKFill

Set CYMK color for filling operations.

Syntax

```
void SetCMYKFill(
    double c,
    double m,
    double y,
    double k);
```

6.10.6 SetCMYKStroke

Set CMYK color for stroking operations.

Syntax

```
void SetCMYKStroke(
    double c,
    double m,
    double y,
    double k);
```

6.11 XObjects

6.11.1 ExecuteXObject

Paint the specified XObject on the area specified by PdfContents::Concat.
On first format, specify name parameter to the name spesified at PdfDoc::AddXObject.
On second format, specify the pointer to PdfXObject directly.

Syntax

```
void ExecuteXObject (
    const char *name);
void ExecuteXObject (
    PdfXObject* xobject);
```

CHAPTER 7

Copyright Permission

HARU takes the ZLIB/PNG license. The license is discribing bellow.

CHAPTER 8

Bibliography

8.1 PDF Resources.

·Adobe Reader (Acrobat Reader).
URL: http://www.adobe.com/products/acrobat/readermain.html

8.2 Programing Resources

·PDF References are available at the following
URL: http://partners.adobe.com/asn/developer/acrosdk/docs.html

·Charsets Index is available at the following
URL: http://www.kostis.net/charsets/

8.3 Type1 Font Resources

·Adobe Type 1 Font Format
URL: http://partners.adobe.com/asn/tech/type/opentype/otover.jsp

·Ghostscript 6.0 fonts
URL: http://sourceforge.net/projects/gs-fonts/

·Type1 KOI8-R font set
URL: ftp://ftp.kapella.gpi.ru/pub/cyrillic/psfonts/

·CM-Super font package
Copyright © 2001, 2002 Vladimir Volovich <vvv@vsu.ru>.
URL: http://www.ctan.org/tex-archive/fonts/ps-type1/cm-super/

·TTF2PTF (True Type Font to Type 1 Font Converter)
URL: http://ttf2pt1.sourceforge.net/

8.4 Copyright Notice

·The IJG JPEG LIBRARY
Copyright © 1991-1998, Thomas G. Lane.

·The PNG Reference Library
Copyright © 1998-2001 Glenn Randers-Pehrson

·The Zlib compression library
Copyright © 1995-2002 Jean-loup Gailly and Mark Adler

·arc4.h,arc4.c (ARC4 crypt)
Copyright(c) 2003 Markus Friedl <markus@openbsd.org>

·Adobe, Acrobat are registered trademarks of Adobe Systems Incorporated.