Rendering Nepali in Linux

Paras Pradhan, Pawan Chitrakar, Srishtee Gurung, Minal Koirala and Sarin Pradhan
MPP MPP MPP KU KU
{paras, pawan, srishtee}@mpp.org.np, minalkoirala@yahoo.com

Abstract

This research document for both Rendering Engine for Nepali in Linux and updating rendering engine for Nepali text are merged together. This is done because as research work on rendering engine was on progress, it came to our knowledge that most of the research work on both the topics was similar and correlated with each other.

This document includes architecture of Pango and implementation of Pango is discussed in details together with a discussion on different Types of Fonts. For rendering in Open Office, ICU is also studied and documented. Last but not the least Since Free Type 2 and Pango helps in rendering a character in Devanagri, Free Type 2 is also conferred in this document.

The actual implementation for rendering Nepali text is included in detail and a comparative study is also given between different versions of Pango for Nepali text. The screen shots using Pango 1.2.0 and Pango 1.4.1 is shown in the report NLCP/PANL10n/Tech/RS/Rend/RPT 03.

1. Pango

Pango is a library for displaying properly internationalized text such as Devanagari, Cyrillic and others. It works on top of several display systems like X fonts, client side open type fonts etc [4]. In this regard, rendering internationalized text involves the Pango engine picking up fonts and finally displaying the fonts. For this, we would require the Unicode Charset or slot for the specific language in the Unicode Charset table. The rendering process also deals with directions of the text and the right glyph selection for a particular script.

In Devanagari script, the characters make up a syllable interacting in complex ways to produce the final rendered form [4]. This type of rendering necessarily involves character combination, cluster formation and reordering. Pango results in wrapping lines, text layout, choosing glyphs and rendering thus acting as the generic model for lying out and displaying internationalized text .

2. Architecture

Several factors underlying the design of Pango are [4]:

- The system uses Unicode as a common character throughout.
- The code for each language is dynamically loaded separately by Pango thus exhibiting the modularity approach.
- Pango language module is further divided into submodules:

    a. Language module, which is system independent in terms of rendering and which deals with tasks like drawing and printing using X fonts.

    b. Shaper module, which is also system independent like the language module and basically dealing with glyph positioning.

- Pango has an abstract class named “Pango Font” . This determines the metrices fo an individual glyph as well as of the font metrices, indivdual glyph metrices and also looks into the coverage of Unicode characters by a particular font. “PangoXfont” subclass handles Xfonts.

- Pango Layout object is the higher level abstraction class. This class gets initialized with [4]:

  a. A Unicode text block

  b. Attributes for the text (font family, size, color, Line-width, line spacing, indention etc). Pango layout deals with interactive editing also such as Cursor movement with arrow keys etc).

3. Rendering of Nepali text

Rendering includes [4] :

- Itemization
- Boundary resolution
- Shaping
- Line breaking
- Rendering

3.1. Itemization

The input Unicode string decomposed into items or tokens and is analyzed by the language module in a single direction [4]. If font's size and style is also set, then the item is again subdivided.

3.2. Boundary resolution

In this step, the word boundaries and line breaks are determined [4]. The function used is: pango_break ().

3.3. Shaping

In this step, characters from each item are converted into glyphs [4]. Function used is: pango_shape ()

3.4. Line breaking

This step makes use of the results of shaping and boundary resolution to choose where to break lines that would require wrapping [4]. Shaping is also called after line breaking if breaking lines involves dividing items.

3.5. Rendering

Rendering is essentially the result of the two processes, viz., shaping and line breaking. The result is a set of glyphs strings (glyphs from the font). For rendering X fonts, libpangox is used whereas for True Type and Postscript fonts libpangoft2 is employed taking the help of the free type library [4].

4. Implementation

Programming Language C has been used to develop Pango . Gobject has been used for Pango development which is found in the older versions of GTK+ toolkit. Pango and glib can be downloaded from http://gtk.org [4].

4.1. Installing GLIB

To install glib, you would need to serially run the following commands:

- ./configure --prefix=/usr
- make
- make install

4.2. Installing Pango

To install Pango, you would need to serially run the following commands:

- ./configure --prefix=/usr --sysconfdir=/etc
- make
- make install

5. ICU by IBM

ICU (international components for Unicode) is the set of C/C++ and java libraries for Unicode support. Before, ICU was the internalization API of JDK 1.1 and later on it turned as the most advanced Unicode/i18n support. The latest version of the Unicode standard is supported by ICU . The same results in terms of rendering are achieved across all platforms using ICU[3].

5.1. Features of ICU

- Unicode text handling.
- Locale and Resources

The ICU package consists of the locale and resource bundles along with the classes implementing them. In addition to this, it contains the locale data and makes the APIs available to access and make use of that data in various services [3].

A locale is the identification of a group of users having similar cultural and linguistic sharings and their general expectations in how their computers interact with them while processing data. This abstract concept is expressed by one of the following points listed below [3]:
:

- Language specification
- Region or country specification.

The language and region specification is specified by a locale ID. In turn the software gets enabled to provide support for culturally and linguistically suitable information. The C++ locale class are provided by ICU with locale IDs [3].

Locale-specific data is stored by ICU in resource bundles. Hence, a general mechnaism is provided to access strings and other objects for ICU services. The **ResourceBundle** contains the locale data in C++. In case of C, this fetaure is provided by the user_interface. ICU provides the generic resource bundle APIs to access these bundles and at the same time facilitates tools to build them [3].

6. FreeType 2

FreeType 2 is a software font engine and designed for high quality glyph image. It is efficient, customizable and portable enough and works well for the text image tools, font conversion tools, display servers etc.It is written in industry standard ANSI C and compiles well with any C compiler [3].

APIs are not provided by FreeType 2 to perform higher level features like text layout or graphic processing. However, it has a simple, easy to use and uniform means to access the contents of the font files[3,11].

6.1. Features of FreeType 2

FreeType 2 provides a simple and easy-to-use API to access font content uniformly, irrespective of the file format. It has the support for scalable font formats like TrueType fonts or TrueType 1. The design is made in such a way that even new glyph image formats are supported and that there is also the facility for glyph manipulation [3]. By default, the following font formats are supported by FreeType 2 [6]:

- TrueType fonts
- Type 1 fonts
- CID-keyed Type 1 fonts
- CFF fonts
- Open Type fonts (both TrueType and CFF variants)
- SFNT-based bitmap fonts
- X11 PCF fonts
- Windows FNT fonts
- BDF fonts (including anti-aliased ones)
- PFR fonts
- Type42 fonts (limited support)

Free Type 2 can read directly from ROM. Thus, the client applications can provide their own memory manager and i/0 stream implementation. The FreeType 2 code also may be compressed just by compiling the modules of the required portions [3].

7. FreeType 1(Ft1) Vs FreeType 2(Ft2)

- Ft1 only supports true type format while Ft2 supports many more formats.
- Ft2 API is more powerful compared to Ft1.
- Ft1 supports Open type text layout processing.

8. Fonts

Fonts are typefaces for screen display and printer output. They represent a graphical design applicable to all alphanumeric symbols in the alphabet. These fonts may come in different sizes and styles. Different types of fonts are [7,8]:

8.1. Bitmap fonts

Bitmap fonts are matrices of dots. Two types of bitmap fonts are [7]:

- Bitmap printer fonts (e.g.: pk)
- Bitmap screen fonts for use in X windows and console (e.g.: bdf, pcf).

8.2. Type 1 fonts

Devised by adobe and supported by adobe's postscript standard. It is distributed as: afm (adobe font metric) or pfm (postscript fonts for windows) and outline file as pfb (printer font binary) or pfa (printer font ASCII).Outline file contains all the glyphs and metric file contains the metrics [7].

8.3. Type 3 fonts

Distributed same as type 1 but not supported by X. Only supported by Postscript standard.

8.4. True Type fonts

Developed by Apple and stores the metric and shape in a single file (.ttf file) [7].

8.5. Type 42 fonts

Same as TrueType fonts in addition to the headers with support for rendering by a postscript interpreter [7].

8.6. Open Type Fonts

Open Type Font, a new cross-platform font is a joint collaborative effort by Adobe and Microsoft . The two main benefits of the Open Type format are as follows [10, 11]:

- It provides cross-platform compatibility i.e. the same font file works on more than one operating system environment.
- It supporta widely expanded character sets and layout features .

9. Use of Pango Engine under Debian

Pango is built for handling rendering of internationalized text under GNU LINUX and since it is modular lots of shaper modules are already available with Pango as: pango-thai, pango-indic etc.

So we have tested Pango for rendering of Devanagari script support. Before, we used Pango 1.2.0 and after the release of Pango 1.4.1 stable version we have created the Debian packages of pango-1.4.1 and its dependent package Glib as libpango_1.4.1-1_i386.deb and libglib_2.4.5-1_i386.deb. Then we installed: Glib and Pango respectively by using the commands:

- ./configure --prefix=/usr
- make
- make install

- ./configure --prefix=/usr --sysconfdir=/etc
- make
- make install

Glib and Pango were installed in Debian linux in which are working on and also checked with few other flavors of Linux and found the rendering is better as compared to Pango 1.2.1.

9.1. Comparative Study

Previously when we worked with Pango 1.2.0 with Red hat Linux there were glyph positioning problems. However, when we used Pango 1.4.1 installed in Debian, there were no such problems. We even checked with different open type fonts, and we found out that the glyph positioning has improved a lot with Pango 1.4.1. Fig-1.1 and Fig-1.2 show difference between the rendering with Pango 1.2.0 in Red hat Linux and Pango 1.4.1 installed in Debian (the glyph positioning problem is visible in Fig-1.1) Fig-2 and Fig-3 are the screen shots of GNOME Desktop and the date using Pango 1.4.1. These screen shots can be referenced in the report NLCP/PANL10n/Tech/RS/Rend/RPT 03.

10. Conclusion

As Pango supports Devnagari Script rendering, there is no need to modify the existing Pango rendering engine. Now, after the stable release of Pango 1.6 we will be checking the major changes and test it.

11. References

[1] O. Taylor. Internationalized Text Handling, In the Proceedings of "*Ottawa Linux Symposium"* 2001

[2] "GTK+ The GIMP Toolkit" http://gtk.org

[3] http://icu.sourceforge.net/docs/papers/icu_overview_latest.doc

[4] http://old.lwn.net/2001/features/OLS/pdf/pdf/pango.pdf

[5] http://freetype.org

[6] http://docs.gimp.org/en/ch06.html

[7] http://www.corel.linux.com/howtos/Font-HOWTO/fontech.shtml

[8] http://www.cdnamesearch.com/html/upload/CDNS%20Guide(11).pdf

[9] http://www.kellogg.cc.mi.us/infotech/grde/courses/grde105/week6.html

[10] ttp://www.mcnaughton-gunn.com/prepress/fonts.html

[11] http://people.freedesktop.org

[12] http://tdil.mit.gov.in

*NOTE: This work has been supported through PAN Localization Project (*www.PANL10n.net*) grant from the International Development Research Center, Ottawa, Canada, administered through Center for Research in Urdu Language Processing, National University of Computer and Emerging Sciences, Pakistan.*