MFCONFIG : METAFONT plug-in module for Freetype rasterizer

Jaeyoung Choi, Sungmin Kim, Hojin Lee and Geunho Jeong

Abstract

One of the advantages of METAFONT is its ability to show a variety of font styles by changing of the values of the parameters that represent the font characteristics. This advantage can be applied to not only simple Roman-alphabet characters, but also to complicated CJK (Chinese-Japanese-Korean) characters. Second, the font families like bold, italic, and bold-italic do not need to be created for METAFONT, because it can automatically generate a variety of styled fonts through changing the parameter values. Therefore, METAFONT can reduce the development time and cost for the production of a font family. It is not possible, however, to directly use METAFONT in general font engines, as it must be changed to the outline-font format if it is to be used in the current PC environment. In this paper, the *MFCONFIG* module that enables a direct usage of METAFONT on Linux is proposed; although, it must be installed with the popular rasterizer Freetype. *MFCONFIG* is a plug-in module for the *FONTCONFIG* library engine, which makes the Freetype engine compatible with the current digital font types of bitmap and outline; furthermore, with the use of different parameters, the proposed module supports a variety of fonts, generated from METAFONT with different parameters.

1 Introduction

Text is an effective way to communicate and record information. With the growing use of smart devices, digital fonts are more commonly used than analog fonts. Although many styles of digital fonts have been created, they still do not meet the requirements of all users, and users cannot change digital-font styles freely[1]; for instance, if a user wants to use a thinner outline font, either he/she has to find a thinner styled font, or an in-application function to change the font thickness. As several different features of font style are needed, though, a simple searching or the changing of the font style of an existing font is not typically easy. A perfect application for the satisfaction of users' diversified requirements regarding font styles does not exist. Also, it is impossible to provide all of the styled fonts in accordance with users' preferences.

Currently, popular digital fonts of bitmap or outline have a limit to change font style[2]. However, METAFONT is a structured font that allows users to change the font style freely. METAFONT, a TEX font system, had been introduced by D. E. Knuth[3]. It has functions for drawing characters and parameters to determine the font styles. When the user changes the parameters, the font style is changed automatically. Therefore, a variety of styled fonts can be generated from one METAFONT font. Figure 1 shows a variety of styled fonts are created by the changing of the thickness, slant, and a combination of two thickness and slant styles for the alphabet "A" and the Chinese character "漢" is shown. If other features such as serif and pen are applied together, a variety of styled fonts can be more generated.

Most users, however, are unable to use METAFONT on their PCs because the current font engines do not support METAFONT. METAFONT is expressed as program code, so it is different from the general digital-font types of bitmap and outline. If a user wants to use a specific METAFONT in general font engines such as Freetype, then he/she needs to convert the METAFONT font into the corresponding outline font format.

In the case of Roman characters, the design of only several hundreds of characters is required, and their shapes are simpler than those of CJK (Chinese-Japanese-Korean) characters. In the mid-1980s, when METAFONT was introduced, the PC was not fast enough to enact a real-time conversion of the METAFONT fonts into the corresponding bitmap or outline fonts. Moreover, outline fonts have been

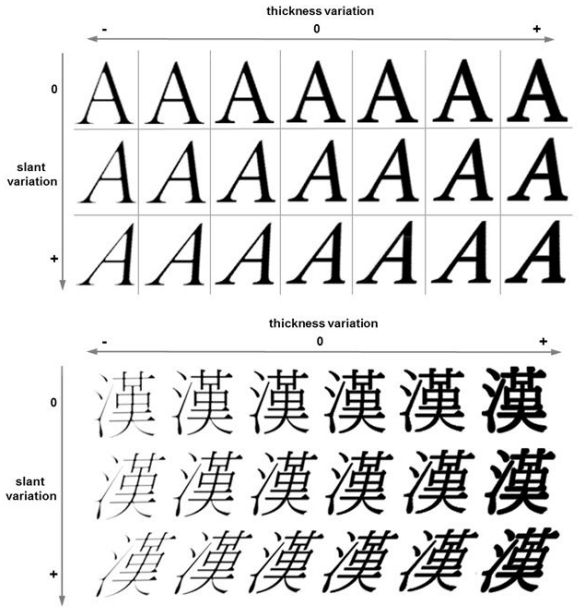

Figure 1: METAFONT style variation

more commonly used, rather than METAFONT, up until the present time.

The current PC, however, ensures good performance regarding the real-time execution of METAFONT. If METAFONT could be used directly in a PC, then users could easily make and use a variety of styled fonts by themselves. As previously we viewed, one METAFONT font can basically be represented by a variety of styled fonts by changing of the style parameter values. Therefore, METAFONT can save great amounts of time and repeated effort in terms of font design to make font families of plain, italic, bold, and italic-bold fonts. In particular, in the case of CJK-character usage, METAFONT could represent an effective way to make and display a variety of font style. Because, compared to the alphabet, CJK characters are very complicated in shape and they are expressed by combinations of radicals.

In this paper, a METAFONT module that enables a direct METAFONT usage on Linux is proposed. It is possible to plug this module into *FONTCONFIG* for providing digital font information to the FreeType engine. If the *MFCONFIG* module is used, the conversion of a METAFONT into its corresponding outline font is unnecessary. It is very simple to change font styles by applying new parameter values. Also, this module can interact with most of the existing *FONTCONFIG* functions without the modifying itself or the Freetype rasterizer. The *MFCONFIG* module therefore has good usability and compatibility regarding the support of METAFONT in the Freetype engine.

2 Research for font system

FONTCONFIG[4] provides an extended font configuration to the Freetype rasterizer, and the Xft(X-FreeType interface library)[5] had been developed to provide interfaces between applications and Freetype. These font libraries are able to collect fonts' information on the current PC system such as fonts' path, fonts' style information, and extra meta information, and so on. Figure 2 shows a font-output sequence that is required from the applications on the X Window system in Linux. After an application sends a font request according to name and style to the Xft library, it also delivers the request information to *FONTCONFIG*. *FONTCONFIG* uses its internal commands to check the following conditions: (1) Whether the requested font is installed, (2) whether the style of the user's request has been applied to the stored font, and (3) whether the requested font has already been stored with the printing format in the cache. (4) If the requested font is not stored in the cache, it needs to be converted into the requested printing format and stored in the cache, and (5) the requested font in the cache is selected and then delivered to Freetype. Lastly, the requested font is printed with the font styles.

FONTCONFIG is a library for Freetype, and it is capable of supporting general digital-font formats that can also be processed in Freetype. The architecture of *FONTCONFIG* is shown in Figure 2. It can support TrueType, OpenType, Type1, CFF, PFR, and DBF, but it does not yet support METAFONT. For the direct support of METAFONT in *FONTCONFIG*, it might be necessary to change the internal codes of *FONTCONFIG*. For instance, the changing of the overall processes in *FONTCONFIG* from fc-scan" (for font searching) to fc-pattern (for the matching of the styled font pattern). It is not a simple work, however. The Xft library interface exists between an application and *FONTCONFIG*, and it is for the providing font information such as font name and font size. It is not a good approach to modify Xft support METAFONT. It might reduce Xft's performance.

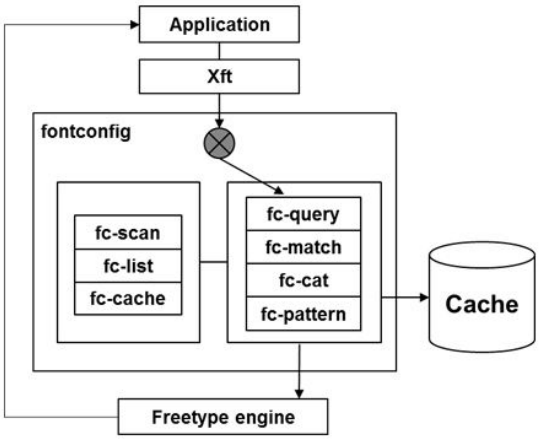

Figure 2: Architectures of fontconfig

VFlib[6][7] is a font driver system for the supporting variety of font types. The system supports virtual fonts like BDF, PCF, and TrueType, as shown in Figure 3. It provides database including general fonts' type information, and it also provides a useful API for the supporting variety of font types. VFlib includes separate modules for each font type, so for the support of METAFONT, a new module can be added to it. But VFlib is heavy because it consists of many different kinds of font drivers and an information dataset of default font information. In addition, the VFlib interface is required if an application wants to use the VFlib library. Therefore, if a METAFONT module is added to VFlib, additional functions must be implemented for every relevant

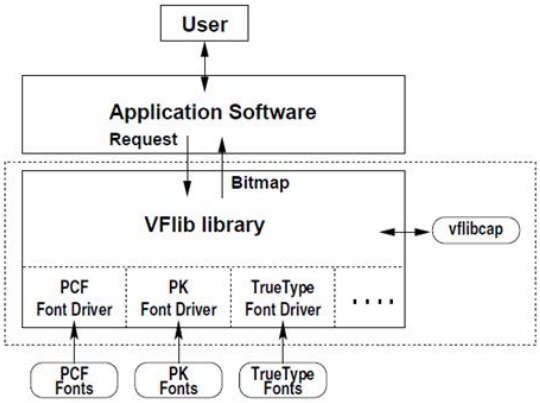

Figure 3: Architectures of VFlib

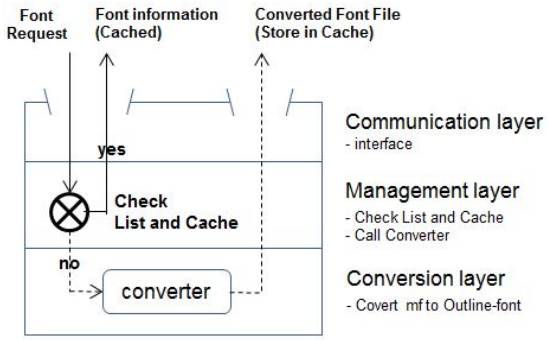

Figure 4: Three layers of *MFCONFIG* module

application. So, VFlib is not suitable to support for METAFONT.

The proposed *MFCONFIG* module in this paper combines the following two features: (1) The process for the printing of digital fonts in *FONTCONFIG*, and (2) the font driver architecture of VFlib. The module can process METAFONT independently, and it can be easily installed or removed since it is implemented as a plug-in module. Also, the steps that are used for its implementation are similar to *FONTCONFIG*'s inner commands, so METAFONT can be used with the existing digital font formats.

3 Implementation of *MFCONFIG* module

As shown in Figure 4, the *MFCONFIG* module consists of the following three layers: Communication Layer, Management Layer, and Conversion Layer. The Communication Layer provides an interface between *FONTCONFIG* and *MFCONFIG*. The Management Layer checks whether the requested METAFONT is ready in the cache. If not, it sends a request message to the Conversion Layer to convert the METAFONT. The Conversion Layer makes a new outline font file by using the requested METAFONT and the customized style values. The resulting outline font is stored in the cache.

As shown in Figure 5, *MFCONFIG* can be plugged into *FONTCONFIG*. First of all, an application requests a font from the Xft library (step 1). Next, *FONTCONFIG* sends the font information and the values of the style parameters to *MFCONFIG* through the interface of the Communication Layer (step 2). This interface checks if the requested font is a METAFONT font or not.

In the case of METAFONT, *mf-query* analyzes the requested information, and *mf-match* tries to find this METAFONT from *mf-list*. If the information does not exist in *mf-list*, the requested METAFONT font is not installed. In this case, *mf-query* returns a not found flag to *FONTCONFIG* (step 3). Otherwise, if METAFONT is installed and is already stored in the cache, *mf-query* returns a found flag to *FONTCONFIG* (step 3).

Sometimes, the requested METAFONT font is installed, but the corresponding outline font may not be stored in the cache memory yet. In this case, *mf-converter* in the Conversion Layer needs to convert the METAFONT font into the corresponding outline font (step 2-a). In this step, the METAFONT font and the styled parameter values from the application are required for the conversion. After the conversion, the outline font is stored in the cache (step 2-b), and *mf-query* sends the found flag to *FONTCONFIG* (step 3).

After step 3, the remaining steps are the default steps of *FONTCONFIG*. The font information is sent to the internal programs of *FONTCONFIG* (step 4) that try to find the corresponding outline font in the cache (step 5); then, this outline font is sent to the Freetype rasterizer (step 6). If *MFCONFIG* returns the not found flag, then *FONTCONFIG* uses a default font file. Lastly, the Freetype engine prints the outline font that is made from the requested METAFONT with the styled parameter values.

The details of the three layers in *MFCONFIG* are presented below. The Communication Layer that is an interface between *FONTCONFIG* and *MFCONFIG* is the starting point of the *MFCONFIG* module. Therefore, Freetype engine receives the font information from *FONTCONFIG* as like previously. The main functions of the interface are as follows: (1) The delivery of the requested METAFONT information to the Management Layer, (2) the returning of the results to *FONTCONFIG*, and (3) the storage of the outline-font file from *mf-converter* in the cache memory.

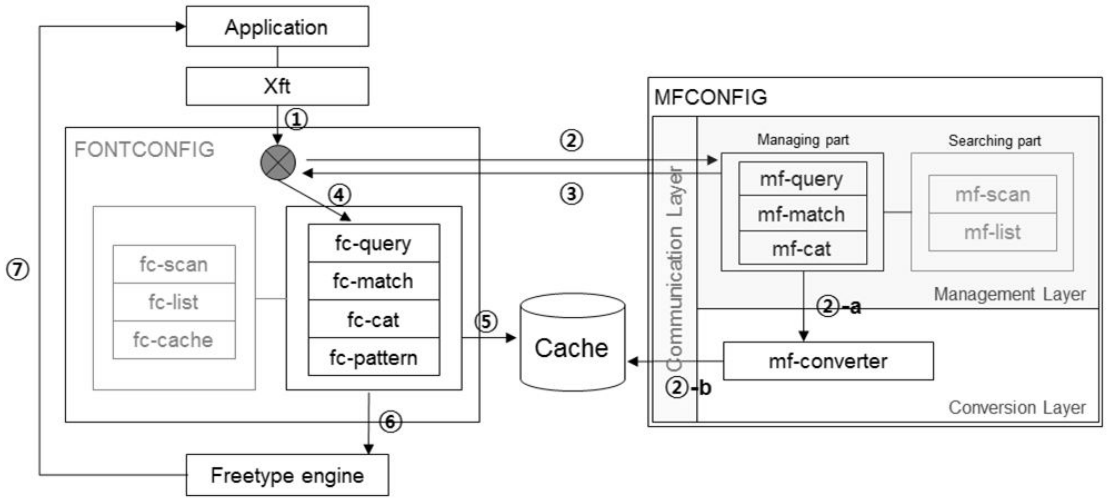

Figure 5: *MFCONFIG* architecture linked fontconfig

The major programs of the *MFCONFIG* module are operated in the Management Layer. This layer is in charge of searching and managing. Searching is an independent function of finding all of the installed METAFONT fonts, and the storing the information in a list beforehand. This list is used for checking whether a specific font is installed or not, and for fetching its information quickly. The searching is implemented with *mf-scan* and *mf-list* that, as shown in Figure 6, work similarly to *fc-scan* and *fc-list* in *FONTCONFIG*, respectively.

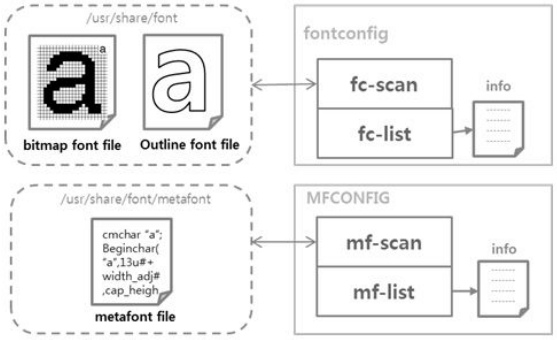

Figure 6: Management layer referenced fontconfig

Management is a core process of the *MFCONFIG* module that is responsible for the following actions: (1) Checking if the requested METAFONT font is prepared in the list, (2) checking if the corresponding outline font is stored with the requested style that is applied to it in the cache memory, and (3) if the outline font is not stored, check whether it needs the conversion of the METAFONT font into the corresponding outline font. If the outline font has already been prepared in the cache, then a notification is sent directly from *MFCONFIG* to*FONTCONFIG* in order to use it. *FONTCONFIG* sends the outline font that is in the cache to the Freetype engine. If the font is not stored in the cache, then the Conversion Layer converts the METAFONT font into the corresponding outline font by applying the style parameters, as shown in Figure 7. The resulting outline font is then stored in the cache, and a notification from the Management Layer through the Communication Layer commands *FONTCONFIG* to use the font.

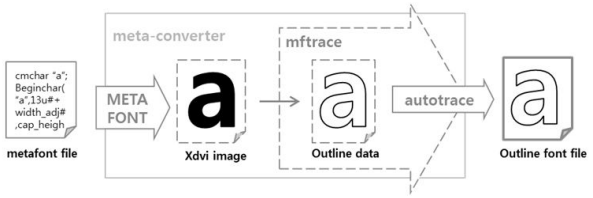

Figure 7: A process of Conversion layer

The work of the *MFCONFIG* module is perfectly compatible with the basic *FONTCONFIG*, and it can append new functions to support of METAFONT. This module works for the management of METAFONT fonts and convert them to the corresponding outline fonts in real-time. When a different style of METAFONT font is requested, *MFCONFIG* can display the resulting font on the screen very conveniently by simply applying the style values to the METAFONT fonts. Therefore *MFCONFIG* has a good usability for METAFONT. In addition, it is not necessary to generate the font-family sets of plain, bold, italic, and italic-bold in advance with respect to *MFCONFIG*,

because a variety of font styles can be generated easily by applying the style values.

Styles	Output	Font files
Normal	Computer	FreeSerif.ttf
Bold	**Computer**	FreeSerifBold.ttf
Italic	*Computer*	FreeSerifItalic.ttf
Bold+Italic	***Computer***	FreeSerifBoldItalic.ttf

Table 1: Print out 4 files of *'FreeSerif'* font family

Style (variable)	Style 1	Style 2	Style 3
Normal	Computer (basic)	Computer (width x2)	Computer (width / 3)
Stroke (hair, stem, curve)	Computer (+20,+10,+10)	Computer (+30,+10,+10)	Computer (+20,+20,+20)
Slant (slant)	*Computer* (1/4)	*Computer* (1/2)	*Computer* (1/3)
Stroke + Slant (slant, Hair, Stem, Curve)	*Computer* (1/4,+20,+10,+10)	*Computer* (1/2,+30,+10,+10)	*Computer* (1/3,+20,+20,+20)

Table 2: Various style fonts using *'Computer Modern'* typed METAFONT with changing style variable

Type	FreeSerif	Computer Modern
(a) Normal	15 ms (10~30)	70 ms (50~80)
(b) Bold	18 ms (10~30)	85 ms (70~100)
(c) Italic	16 ms (10~30)	105 ms (70~110)
(d) Bold+italic	16 ms (10~30)	100 ms (90~120))

Table 3: Average time for printing out 2 different fonts types (millisecond)

4 Examine *MFCONFIG* module

For the performance of the experiments of this stud -y, an application for the use of the X Window system in Linux was developed, and the display of a text file was attempted with the use of a variety of font files. In addition, the TrueType font family named *"FreeSerif"* is used along with the four font styles (*normal, bold, italic*, and *bold+italic*) and a METAFONT font named *"Computer Modern"* were used. The *"FreeSerif"* font family consists of the following four files: *FreeSerif.ttf, FreeSerifItalic.ttf, FreeSerifBod.ttf,* and *FreeSerifBoldItalic.ttf*. Similarly, the *Computer Modern* font was examined along with the four styles *normal, thickness, italic*, and *thickness+italic*. The sample text comprises over 2,000 words and over 8,800 characters, including the space characters. For the performance analysis regarding the Freetype rasterizer, the time between the requesting of a font with styles from an application and the successful display of on-screen text were measured and compared.

Table 1 shows the *"FreeSerif"* font family with the four different styles, and Table 2 shows 12 styles that were generated for the *"Computer Modern"* font of METAFONT. All of these styles were made from one original prototype of the METAFONT font by simple changing of the style parameters. Therefore, the METAFONT font has good capability of generating various font styles with the style values.

For the printing out of a text file on the application, four of the *"FreeSerif"* files from Table 1 and Style 1 of *"Computer Modern"* from Table 2 were used. For Style 1 from Table 2, the four parameter values are *hair, stem, curve, and slant*. The three parameters of *hair, stem*, and *curve* are related to the *bold* style, but these parameters are different for lowercase and uppercase. The *slant* parameter is related to the *italic* style. The chosen parameter values for the representation of a bold style are hair+20, stem+10, and curve+10, while the *slant* is 0.25 for a representation of the *italic* style. Figure 8 shows the results of the printing of *"FreeSerif* with four different styles, and Figure 9 shows the results of the printing of the *"Computer Modern"* font.

Table 3 shows the average time to print out of both the *"FreeSerif"* and *"Computer Modern"* contents. In this experiment, the results from 10 ms to 30 ms were obtained, and the average time is 16 ms, while four TrueType fonts were used. Therefore, extra time was required for the conversion of the TrueType fonts. In the case of the *"Computer Modern"* font of METAFONT, the result is much slower than that of *"FreeSerif"*, because it needs additional time for the conversion of the METAFONT font into the corresponding outline font. The obtained results are from 50 ms to 120 ms, and the average time is 90 ms. Even though this time is 10 times slower, 90 ms is still a short time for the printing of a font file on screen. It is possible to conclude that the *MFCONFIG* module could be used with *FONTCONFIG* for the support of METAFONT on a Linux PC almost in real time.

The *MFCONFIG* module is a convenient system to provide users with various styled fonts on screen by applying style parameters directly to the METAFONT font. The users can use METAFONT easily like TrueType font as shown in Figure 9.

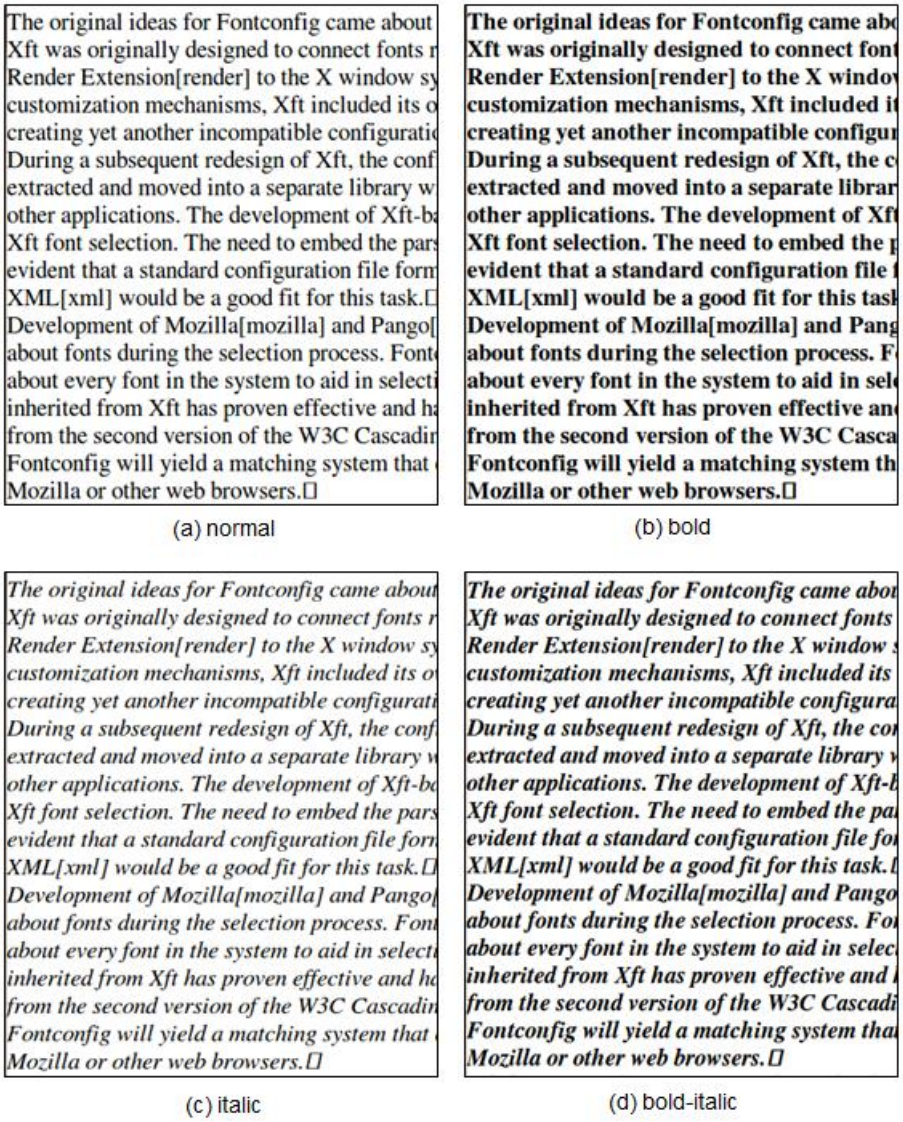

Figure 8: Print out text with four styles of *"FreeSerif"(normal, bold, italic, bold+italic)*

In this paper, the METAFONT font *"Computer-Modern"*, which provides alphanumeric values and symbols, is examined. It is possible to perform tests with the other METAFONT fonts from CTAN (comprehensive TeX archive network) directories that support languages such as Russian and Thai. But difficulty was experienced regarding the testing for which complicated CJK fonts are used.

CJK fonts are very complicated as compared to the alphabet-based fonts and they are composed of several thousands of phonemes. A number of studies have been conducted to partially implement the CJK fonts, such as Hongzi[8][9] and Tsukurimashou[10], including the use of a structural font generator using METAFONT for Korean and Chinese[11], and so on. However there is no commercialized level of files has not yet been created by METAFONT for the CJK fonts. The authors expect that the use of the *MFCONFIG* module for the generation of the CJK fonts will take more time. It may, however, be possible to solve this problem by optimizing metaconverter in the Conversion Layer. Currently, metaconverter works with *"mftrace"* and *"autotrace"* programs, which takes a long time to generate outline fonts.

5 Conclusion

In this paper, the *MFCONFIG* module, which enables the direct use of METAFONT on Linux, is proposed. It is installed and used with the popular Freetype rasterizer. *MFCONFIG* is a plug-in module for the *FONTCONFIG* library of the Freetype engine, and

Figure 9: Print-out text with "*Computer Modern*" font in four styles (*normal, bold, italic, bold+italic*)

makes the latter compatible with the current digital font types of bitmap and outline. The module supports a variety of the styled fonts that are generated from METAFONT along with different parameters.

The existing digital fonts - mainly the outline fonts of Type1 or TrueType - either do not allow users to change their styles, or they can be changed within very limited ranges such as font size. From the experiments of the present study, however, it has been demonstrated that a variety of fonts can be directly generated on screen by applying different style parameters to a prototype of theMETAFONT font on a Freetype rasterizer that is installed with *MFCONFIG*; furthermore, the fonts could be seen within an average time of 90 ms, which is a barely noticeable duration.

MFCONFIG module targets METAFONT fonts to be used with Freetype which is a famous rasterizer. *MFCONFIG* could be used effectively in case of alphabet-based fonts, which are relatively simple and have a limited number of characters. However, there are only a few METAFONT fonts for various languages. It might take a longer time to process CJK METAFONT fonts, which have complicated shapes and have more than several thousands of phonemes. Additional works will be focus on these CJK METAFONT for improving performance, and try to optimize of *MFCONFIG* module. In addition, this module will be experimented such like one of font driver in the Freetype rasterizer.

Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education(2015R1D1A1A02062211).

References

[1] S. Song. Development of Korea Typography Industry. *Appreciating Korean Language*, 2013.

[2] Y. Park. Current status of hangul in 21th century. *Type and Typography magazine The T*, 7th.

[3] Donald E.Knuth. Computers and typesetting volume c : The METAFONTbook. *TUGboat*, 1986.

[4] K. Packard. Fontconfig. *Gnome User's and Developers European*, 2002.

[5] K. Packard. The xft font library: Architecture and users guide. *XFree86 Technical Conference,Citeseer*, 2001.

[6] H.Kakugawa. Vflib- a general font libray that supports multiple font formants. *EuroTEX confrerence*, March 1998.

[7] H.Kakugawa. A general purpose font module for multilingual application programs. *SP&E*, March 2000.

[8] JR. Laguna. Hong-zi: A chinese metafont. *Communications of the TEX Users Group*, 2005.

[9] Candy L. K. Yiu, Jim Binkley. Qin notation generator. *TUGboat*, 26th, 2005.

[10] Matthew Skala. Tsukurimashou: A japanese-language font meta-family. *TUGboat*, 34th, 2013.

[11] Gyungjae Gwon, Minju Son, Genho Jeoung, Jaeyong Choi. Structural font generator using METAFONT for korean and chinese. 2016 (in preperation).

◇ Jaeyoung Choi
Soongsil University, Seoul, Korea
choi@ssu.ac.kr

◇ Sungmin Kim
Soongsil University, Seoul, Korea
sungmin.kim@ssu.ac.kr

◇ Hojin Lee
Soongsil University, Seoul, Korea
hojini@ssu.ac.kr

◇ Geunho Jeong
Gensol Soft, Seoul, Korea
ghjeong@gensolsoft.com