உத்தமம்
INFITT
உலகத் தமிழ்த் தகவல் தொழில்நுட்ப மன்றம்
International Forum for Information Technology in Tamil

Penn
UNIVERSITY of PENNSYLVANIA

மாநாட்டுக் கட்டுரைகள்
CONFERENCE PAPERS

TAMIL INTERNET 2011

தமிழ் இணையம் 2011

Open Source Tamil Computing

S. Gopinath and E.I. Nehru
National Informatics Centre, Chennai

Abstract

For many of us English is the natural choice for commodity Computing such as Internet Web browsing, email, Instant Messaging, Word Processing etc. As computer proliferates in every aspects of our daily life, it is a need to have some kind of Multilingual Computing. For example an individual would like to send an email to his friend in a local langauge (like sending a greeting email in Tamil), an e-Governance application needs to be benifited for persons who are not familiar with English, or a product brochure to be made available on Web in a local langauge. More importantly the multi-lingual data should be interoperable and to be long lived across computer systems. The fundamental requirements for the Language Computing are

1. Coding system for Local Language Script Set,
2. Input Methods, Output Methods,
3. Software Libraries, APIs, Fonts
4. Applications

The aim of this article/paper is to summarize the various aspects of Multilingual Computing in OpenSource Plaform with emphasis on Linux and Tamil.

Contemporary Linux systems comes withMultilingual features and in most cases they can be enabled very easily.

1. Unicode

Internally, coding of character system to be done to make meaningful processing of characters. A decade ago Standard ASCII (7bit) is a very popular encoding system for English characters. Later extended ASCII (8 bit/1 byte) is used for representing English characters and several Latin characters, Indic characters etc. But a true multilingual system needs to represent all possible script set (including Math Symbols, braille characters etc.) independently so that any electronic text shall contain all sorts of characters.

In late 1980s, Joe Becker (Xerox), Lee Collins and Mark Davis of Apple conceptualized multilingual character set encoding. Joe Becker published a draft for International/Multilingual Character Encoding System and tentatively called Unicode. This proposed 16 bit character model (each character is to be represented by 16 bits). This proposal is popularly called as Unicode88. Later when Unicode 2.0 was proposed, the character width is no longer restricted to fixed 16 bits and hence its posible to represent many ancient character sets like Egyptian Hieroglyphs. As of 2011 Unicode 6.0 standard prevaills.

Unicode defines codespace containing code points in the range 0x0 to 0x10FFFF (represented in Hexadecimmal) which means there are 1,114,112 code points in the Unicode Codespace. Each character is assigned a code-point. Unicode codespace is divided in to 17 planes and numbered from 0 to 16. The Plane 0 is called ”Basic Multilingual Plane” which contains codepoints numbered from 0x0 to 0xFFFF. Code Points for Tamil Character Set, other Indic Character Set, Cyrillic, Geometrics Shapes, Math Symbols, Arabic etc are allocated in this plane (hence this Plane-0 is called Basic Multilingual Plane). Other planes are

1. Plane 1 - Supplimentary Multilingual Plane
2. Plane 2 - Supplementary Ideographic Plane
3. Plane 14- Supplementary Special Purpose Plane
4. Plane 15,16- Private Area Use

The Tamil Language is allocated codepoints from 0x0B80 to 0x0BFF (128 codepoints) in BMP Plane. Please remember, that the codepoints has nothing to do with physical representation of characters in computer binary bits (or qbits in Quantum Computing). The physical representation also called as Unicode Character Encoding is done through a methodology called Unicode Transformation Format and it is defined by the Unicode Constortium. In Posix Systems and in Internet, UTF-8 Encoding Scheme is popular. In this article/paper, by default we frequently refer UTF-8 encoding scheme.

The UTF-8 was initially proposed for Plan9 Operating System. It is a multibyte character encoding system which uses one byte to 4 bytes depending upon the codepoint. The encoding for the code point from 0 to 127 is same as ASCII encoding for ASCII space 0 to 127 and hence it is already compatible with ASCII for the code points 0-127. The main advantage for the UTF-8 is its self syncronising and does not depends on endianness of the computer system. No special marking in the data stream is required for UTF-8. The main disadvantage is that it needs more bytes and it requires extra processing.

One of the basic question that arises is this. We have only 128 codepoints alloted for Tamil in the Unicode and we have 247 number of characters in Tamil Character Set. Is this a right way ?. How do we manage to represent all characters in Tamil Character Set?. This is very debatable. The brief explanation for the question how to accomodate 247 characters in 128 code point is as follows.

The most of the characters in Tamil are conjunct characters and share the same set of glyphs (or visual repersentation) in most of the characters and they are highly structured. Hence, it is possible to represent the Tamil Characters with very little compromise within 128 code point. But conjunct characters takes more bytes space than vowels or consonants (consonant equivalents).

2. Input System

It is quite obvious that keyboard is the de-facto input device for humans (Yes.. we have voice and other input systems for Humans!).In PC based architecture, keyboard is a raw device, and has no language centric mechanisms expect that in most of the keyboards we have ”English” letters inscribed on keys. One of the questions beginners of Tamil Computing asks, is there any keyboard where I can find all 247 characters ?. The answer is similar to that of in the previous section that we indeed do not require individual keys for all the 247 characters (infact one can make that eiether physical or virtual onscreen keyboards, but it would be expensive and bulky) as many are conjunct characters and many share common glyphs (diacritics) and hence the normal general purpose contemporary keyboard is suffice. This methodology requires some kind of software popularly called Input Methods to assemble the characters when multiple keystrokes are required to form conjucnt characters.

Pioneering works has been done by the creators of m17n libraries which is quite popular in Posix systems. The m17n library is a multilingual text processing library for C language. In Linux the popular Input Methods avilable are Smart Common Imput Method (SCIM) and iBUS (Intelligent Input Bus). Most of the current Linux uses iBUS by default.

3. Output Methods

The whole fanfare in Multilingual Computing is on the output. Unless one sees the output in appropriate script, the multilingual computing is not fullfiled. Its the most obvious part of Multilingual computing. Normally, we talk about fonts whenever we discuss about Multilingual Computing. Fonts contains information about how to draw the shapes on the screen. The renderer engine which part of the output system takes the font and renders the shapes. The fonts are usually distributed in a file (often called as a font file). There are 3 types of fonts. They are Bitmap fonts, Outline fonts and Stroke based fonts. The bitmap fonts or raster fonts contains pixel images of the glyphs. The Outline fonts or vector fonts contains the vector images and mathematical formula can be applied to get the different sizes of a glyph. The popular fonts types such as TrueType, OpenType, Adobe fonts are Outline fonts.

One of the functions of m17n library is displaying and rendering multi-lingual texts. It requires complex processing to render scripts such as Tamil, Devanagiri etc. A character may have a single glyph or few glyphs spaced at one another. A re-ordering may also be required. The technology for such rendering is known as ″Complex Text Layout″ or CTL. The m17n library database contains what is called Font Layout Tables (FLT) which bridges the fonts and the rendering engine.

The present day Linux distribition such as Debian, RedHat are available with TrueType and OpenType Font rendering engine. The Lohit fonts are quite popular in Linux Community. Lohit Fonts Project is sponsored by RedHat and its a Fedora Hosted. CDAC distributes OpenType Fonts for Indic Languages. By default most of the Linux distros has Tamil fonts by default or can be added very easily.

4. Software Libraries/APIs

The GNU C Library (glibc) supports UTF8 encoding and contains functions for multilingual computing. The utilities such as sort, grep which are based on glibc can be used on multilingual texts. The holy grail M17N C library is the flag ship for multilingual computing. The link http://unicode.org/resources/libraries.html

contains links for various multilingual libraries.

5. Applications

The end user experience on multilingual computing based on the Applications. The simple commodity applications like Internet surfing, Word Processing and e-mail etc. can be done using exisiting applications like Mozilla Firefox, ThunderBird, OpenOffice, Libre office etc. The user has to only choose the appropriate Input Method (so that he can type in Tamil). Applications like Yahoo Messenger is multilingual and once can chat in clear Tamil (instead of transliteration) Empathy which is an GNOME application can be used for Instant Messaging and its multilingual. Simple editior like gedit can be used to create multilingual files.

Web Static Pages can be created by including a http header Content Type. This can be done by adding a meta tag between <head> and </head> as follows.

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
</head>
```

In the body section, the content can be multilingual. The multilingual content can be created even with simple editior like gedit. Of course the browser and the system in which it runs must have UTF-8 support with appropriate language fonts installed . Enterprise and backend applications use Database Management Systems and in Open Source Community one of the most popular DBMS is Postgres. The current version of Postgres supports UTF-8 encoding by default on Posix systems. This facilates the storage of data in multilingual form.

6. Requirements and Suggestions

Although it is possible to carry out most of the multilingual computing, the following are the suggestion by the authors for further enhanced functionality.

1. The simpler keyboards can be made to meet the various requirements of the end user. For example to use TamilNet99 keyboard requires practice and considerable understanding of the scripts, the people who just knows only to read or write the scripts could not easily use these keyboard layouts or definitely not as trivial as someone who only knows English and use English keyboard. A much simpler keyboard may be prototyped after doing sufficient study on the above aspects. As the input methods are implemented in a software, these initiatives can be done without changes in underlying software.
2. As of now Input Methods are implemented in software and they run in main system/CPU. It is possible to design and implement an Intelligent keyboard, which directly sends the UTF-8 sequences instead of raw keyboard scan codes. As Unicode integrates across language/scripts, culture etc, the Universal Intelligent Keyboard is appropriate. The keyboard can be made programmable such that the end user can configure the keyboards to function eiether as raw scancode mode, or direct UTF-8 mode. If the same kind of keyboard is mechanically designed foldable/wrappable, it can be used in mobile devices such as smart phones, tablets universally (which can bring down the costs).
3. The authors could see that more works to be done on Optical Character Recognition, Speech to text conversion, Accessibility in the context of Multilingual computing.
4. Better Tanil outline fonts needs to be forged for High Resolution equipments.

7. Conclusion

Multilingual computing in Open Source Systems like Linux is matured. Works like M17N C libraries, UTF-8 encodings really makes Multilingualization possible in Linux like systems which carefully eliminates need for any special magic numbers. Due to glibc and m17n libraries, large number of applications are already multilingual enabled.

8. Acknowledgements

I would like to thank Mr.T.N.C. VenkataRangan of Vishwak Solutions Ltd, for spending hours in answering my novice questions when I started to learn Multilingual Computing.

My special thanks to CDAC Chennai offcials (Mr.C.Sudhakar, Mr.Nagesh, Mr.Gowri Ganesh) for assisting me in learning some of the concepts in this area.

My thanks to my seniors (Mr.E.I.Nehru,Mr.JD Prince) for encouraging as well answering my doubts on the subject. My thanks to NIC's FMG Engineers (Mr.Emthias and others) for their Technical Support.

I thank NIC for providing enough resources.

Thanks to IITM Students for providing help in preparing the document in LaTex.

Not but not least, my thanks to my parents and my brother who patiently accepts my late arrivals and patiently serve late dinner as I had to spend late hours while preparing this article.

9. Few References

- http://www.cl.cam.ac.uk/~mgk25/unicode.html
- http://www.cl.cam.ac.uk/~mgk25/ucs/UTF-8-Plan9-paper.pdf
- http://code.google.com/p/ibus/
- http://www.m17n.org
- http://www.postgresql.org/docs/9.0/static/sql-createdatabase.html
- Ofcourse, Wiki and Google pages.