Tamil Unicode Implementation Issues

Sivaraj D.
Hyderabad, India

Abstract

This paper takes stock of Tamil Unicode support in various platforms and toolkits and discusses implementation issues that still need to be resolved. It enlists the steps needed to achieve reasonable Unicode support. The paper also discusses a few other important internationalisation concepts which may not be dependent on Unicode.

1. Current Status

1.1. Defining support levels

Support for Tamil Unicode can be classified into two wide areas: platform support and application support.

When a platform supports Tamil Unicode, all applications written for that platform will automatically be Unicode enabled. However older versions of the platform may still be lacking. Please note that a platform is not same as an operating system. For instance, GNU/Linux operating systems has multiple platforms, namely, console, Gnome, and KDE. Usually toolkits used by such platforms are where Unicode support is implemented. For our convenience, let us define platform support at two levels, namely basic support, advanced support.

There are some applications which are cross-platform. When such cross-platform applications are Unicode enabled, depending on the method used for achieving the functionality, Unicode support may or may not be available in specific platforms.

If a platform or an application is capable of displaying Unicode text, and it has a well defined mechanism for entering text in Unicode via keyboard it can be considered to have Unicode support at basic level.

Advanced support would constitute the ability to properly sort text in the traditional sort order, ability to determine word boundaries and hyphenation, spell checking, support for localisation of user interface, calendar and currency notations, and probably a thesaurus, grammar checking, and number formatting.

In a distant future support for traditional calendars would also need to be considered.

2. Major Platforms and their status

Tamil support for Unicode is starting to appear in various toolkits under different operating systems and utilities. Notable among them are Uniscribe (from Microsoft), Pango (part of GTK+/Gnome), and QT (from Trolltech).

2.1. Microsoft Windows

Support for Unicode Tamil range in Microsoft Windows platform is achieved through the Uniscribe library. Full support for Unicode for text display and input is available in operating system releases of Microsoft Windows 2000 and later in server operating systems, and Windows XP version of consumer operating systems.[1]

Uniscribe is a relatively mature library which doesn't have any major issues in basic handling of Tamil Unicode.

2.2. Unix platforms

Pango[2] is the first library to bring Unicode support for Indic scripts in Unices, shortly followed by QT[3]. While both these toolkits support Tamil to a great extent, minor problems still exist, mainly in their input mechanisms. Pango supports keyboard input via custom input modules, while QT keyboard input is via standard X protocols. However both Pango and QT appear to be moving towards Internet-Intranet Input Method Framework (IIIMF)[4].

Pango library is mainly used by GTK+ toolkit which is used by Gnome[5] platform, while QT is used by KDE[6] platform. Both these toolkits are available under most Unix and Linux based operating systems, and even Microsoft Windows. However Windows support for Pango is usually lagging behind and not fully supported.

Currently no known Unix/Linux system supports Unicode Tamil in their consoles. While the text rendering is supported by X based terminals like Gnome Terminal, the text is mostly unreadable due to the lack of suitable fixed width fonts.

2.3. Mac OS

Currently Mac OS doesn't have platform level support for Tamil Unicode. However application support is possible for GTK+ toolkit via Pango for Mac OS X and later.

3. Keyboard drivers

3.1. Behaviour of Backspace and Delete keys

Currently behaviour of keyboard drivers in various platforms are inconsistent and in some cases incorrect. The problem is mainly visible in the Backspace and Delete keys. These keys may either delete a single Unicode character, a single Tamil character which may be a combination of a Unicode vowel or consonant character along with modifiers, or a cluster of characters.

MS Windows and QT toolkit implement the first method of deleting a single Unicode character. Pango currently deletes a cluster of characters starting with any preceding *oRRu* character. For example, consider the word *appA*. If the Backspace key is pressed with the

cursor at the end of the word, it will delete *ppA* leaving only 'a'. This behavior is obviously incorrect. The problem occurs since Pango implements Tamil as part of other Indic scripts. In most other Indic scripts *ppA* sequence is a rendering cluster. Presence of *oRRu* (aka *virama*) alters the rendering of the entire cluster.

Action items:

a) Decide which is the correct behaviour - deleting a single Unicode character or a single Tamil character.

b) Fix behaviour of Pango to confirm to the previous decision or at least to follow one of the first two methods.

3.2. Auto-pulli feature

Tamil99 keyboard is widely considered as a standard, and is supported by Tamilnadu Government. This keyboard features a unique idea namely the auto-pulli feature. The purpose of this feature is to reduce number of keystrokes required to type common words. However more often, it also adds problematic and inconsistent behaviour. This feature is also liable to be incorrectly implemented either due the limitations of the keyboard driver software or due to incorrect understanding of the developer.

For example, consider the name *vIrarAkavan*. Here 'rarA' sequence (keystrokes 'm', 'm', and 'q') will trigger auto-pulli feature. To get the 'ra' in second syllable the user has to type an additional 'a' (as 'm', 'a', 'm', 'q') to force *akaramey* instead of *oRRu*.

This feature also increases the learning time of the user. Consider which is easier to learn:

- Type any consonant followed by 'f' key to get *oRRu*, or,

- Learn about situations where the user need not type 'f', and also learn about the exceptions where he needs to type 'a' to get *akaramey* instead of *oRRu*.

Most speed typists will try to type the *pulli* automatically for all pure (stand alone) consonants to avoid thinking for each word. Even for those who are not up to speed, the learning curve is longer with auto-pulli feature than without.

Action items:

a) Decide whether auto-pulli feature is really useful, and,

b) At the least, provide an option in the keyboard driver (or an alternative keyboard driver) without auto-pulli feature, and publish this requirement as part of the standard.

3.3. Other standardisation issues

While Tamil99[7] keyboard defines most of the characters available in Unicode

repertoire, some of the newly added characters and symbols are not yet available. These also need to be added to the standard. Behaviour of certain keys is not fully defined or ambiguous due to limitations of Unicode. These items need to be further clarified.

3.3.1. SRI

After the addition of the new 'SHA' character (U+0BB6)[8] in the Unicode Tamil range, sequence of SRI should be changed to use this character instead of the older SSA character. Suitable changes need to be made in application softwares as well to accommodate this change.

A new keystroke for the new SHA character also needs to be defined.

3.3.2. KSHA

Unicode currently defines the combination of KA + *Pulli* + SSA as KSHA. To render this sequence as two separate glyphs namely K and SSA, a ZWNJ (Zero width non-joiners) need to be inserted between KA + *Pulli* and SSA. The keyboard behaviour can be changed so that it automatically generates this ZWNJ character when this sequence is typed separately, but the basic sequence is generated when the key for KSHA is pressed.

However this change also depends on the efforts to get KSHA character added to Unicode as a separate character. If this is achieved then the current keyboard behaviour can prevail, with suitable changes to accommodate the new KSHA character.

3.3.3. Tamil numerals

Key stokes for Tamil numerals also need to be defined.

3.3.4. Keyboard switching

Currently there are no standard key stokes or sequences defined to switch keyboard layouts between Tamil and other languages. This is left to the discretion of the individual developer. It might be useful to define a specific key stoke for the purpose. In specialized Tamil keyboard hardware (or a trilingual keyboard like English/ Sinhala/ Tamil in Sri Lanka) a separate key may be provided for this purpose.

4. Word wrapping

Currently none of the platforms implement Tamil specific hyphenation behaviour. Word wrapping is only done at the word boundaries at white space. To have a decent support for Tamil text processing, proper hyphenation behaviour is necessary. Preferably this support needs to be added at the toolkit level, so that all applications can take advantage of this facility. In Win32 systems this would be in Rich Edit control. A stand alone library for Gnome platform named libhnj seems to be out of development. It will probably handle by

Pango directly. Hyphenation may also be implemented directly in application programs like Open Office independent of the toolkit.

One major pitfall that should be avoided is development of such hyphenation scheme in combination with other Indian languages. As seen earlier word structures in other scripts are different in Tamil and other Indic scripts. Tamil has the requirement that pure consonants should not appear in the beginning of a line or sentence. This requirement is not present in most other Indic scripts. In fact, it might be preferable to break a word just before a pure consonant in other Indic scripts.

Action items:

a) Develop a hyphenation algorithm based on Tamil grammar.

b) Implement this algorithm in various applications and toolkits.

5. Sorting

5.1 Sorting of specific characters

Standard Tamil sorting sequence is well defined for pure Tamil characters namely the 12 vowels, *aaytam*, and 18 consonants. Problem starts when the *grantha* characters are added to the mix. Different regions and publications within the same region use different sort orders for *grantha* characters.

Sort order for symbols and numerals and their place in the entire order also need to be defined.

5.2 Sorting with multiple scripts

This issue is often overlooked, but gains importance when we consider Unicode. Unicode enables us to mix multiple scripts within a single document or database. So we might have to decide which script should be sorted first. Unicode defines a default script sequence which is mostly based on their location in the Unicode repertoire. Government agencies might set their own standards.

There are need situations where transliterated equivalents may have to be sorted together. For example, in a voter list, a citizen's name may have to be sorted together regardless of the script it is written in. However such are specialized requirements and may be handled by specialized softwares.

6. Website Development

6.1 Use UTF-8

Often times, websites which proclaim to be based on Unicode use it incorrectly. The right way to use Unicode in websites is to use UTF-8 as content encoding. However web pages may define themselves as ISO-8859-1 encoding (or another similar 8-bit encoding).

This often happens when the default encoding in website development tools like Front Page are incorrectly defined. This will result in the pages using HTML entities instead of direct Unicode encoding.

There are many disadvantages to such a scheme. Some browsers may mistreat HTML entities. These are prone to be split at the wrong places and rendered incorrectly. Browsers will be unable to determine the correct font to be used for rendering such text.

So web pages should always explicitly specify their content encoding as UTF-8. An easy way to find out whether your web page contains HTML entities is to open the web page in a text editor like Notepad. The HTML entities will be displayed as 4 digits numbers with an ampersand and semicolon. For example, the letter 'ka' would be displayed as 'க'.

6.2 Dynamic fonts

Even with Unicode, many websites are using dynamic fonts. To make the matters worse, some websites use fonts with non-standard characters as dynamic fonts. Such fonts contain characters that are not familiar to most Tamil readers - like right modifiers for *ekaram, EkAram*, etc - called linear Tamil. Such practice need to be stopped to maintain sanity in Tamil web community. Some people might have started using such fonts to overcome problems in older systems that do not support Unicode. However it should be noted that use of these non standard glyphs will make your website equally non-readable for most Tamil population with these older systems, and the readers who actually have a Unicode supported system as well. If you target audience constitutes large number of people using older systems, you will be better of using an 8-bit encoding like TSCII instead of Unicode.

Best way to express design preference for your website is to use a list of preferred fonts in the style sheet with a *correct* Tamil font listed in the beginning.

7. Mobile and Embedded devices

An embedded device might be a mobile phone with a web browser and email client or as simple as a Television with Tamil user interface.

Using Tamil Unicode requires significant processing power, which may prevent its use in embedded devices. It is possible to develop a simpler version of Tamil Unicode rendering for such devices. However Unicode would still need bigger storage than an 8-bit encoding.

Some points that need to be considered are:

a) Amount of processing power available,

b) Does this application need to coexist with scripts other than English and Tamil?

c) Amount of memory available, and

d) Bandwidth requirements if this device needs to interact with other devices.

8. Conclusion

While this paper discusses various current issues that faced with Tamil Unicode implementation, this is not an exhaustive list. We still have a long way to go even to achieve basic Unicode Tamil compatibility across all platforms. But with persistent effort it is possible to see the light at the end of tunnel.

Bibliography

[1] http://www.microsoft.com/typography/developers/uniscribe/default.htm

[2] http://www.pango.org/

[3] http://www.trolltech.com/

[4] http://www.openi18n.org/ (look for IM(IIIMF) on the left hand panel)

[5] http://www.gnome.org/

[6] http://www.kde.org/

[7] http://www.tamilvu.org/Tamilnet99/keystand.htm

[8] http://www.unicode.org/alloc/Pipeline.html