Free and Open Source Software
Asia-Pacific (FOSSAP) Consultation

9 – 11 February 2004, Kuala Lumpur, Malaysia

Localization of FOSS

Sarmad Hussain
Pakistan

International Open Source Network (IOSN)
An Initiative of UNDP's Asia-Pacific Development Information Programme
http://www.iosn.net

Localization of FOSS

Dr. Sarmad Hussain
Center for Research in Urdu Language Processing
National University of Computer and Emerging Sciences, Pakistan
sarmad.hussain@nu.edu.pk

1. Background

Information has now become such an integral part of our society, that its access is considered a basic human right. Moreover, development of rural and urban developing populations is also getting increasingly dependent upon access to information. This is specifically applicable to Asia which houses the largest developing population. 32 out of 50 Asian countries are considered to be under-developed, having annual per capita income of less than $500. Access to information is thus critical for these countries.

Asians form about 61% of the world population and since 2001 they have also become the largest group of Internet users. Currently, 173 million Asians use the internet, which is 27% of the total internet users across the globe. However, these users form only about 4.5 % of the total Asian population, which shows that there is enormous potential for Internet usage in Asia. In addition to being most populous, Asia is also the most culturally and linguistically diverse region of the world. There are 2197 languages spoken in Asia, which is the largest number of languages spoken in any one region. Only about 20% of these people can communicate in English.

Investments have been put into developing ICT infrastructures in Asia. Nevertheless, the persisting digital divide attests that the current path towards providing connectivity and technology infrastructure alone would not enable the majority of Asian populations to benefit from the present information availability. One significant reason is that these populations cannot circumvent the obstacle of English language content, which dominates the Internet. Unless these large non-English speaking populations have the ability to do computing in local languages, they will not be able to use ICTs for their development effectively. Another important reason is the high cost of proprietary software and expensive hardware needed to run this software (in some countries basic proprietary system for using computers may cost as much as annual per capita income).

The obvious solution to overcome the existing financial and linguistic barriers to access information by Asians is to provide them *localized free open source software*. Localized software would enable them to access, generate and exchange information in a language they can understand. Additionally, FOSS is free software and additionally does not require expensive hardware to operate.

This paper outlines the scope of localization in general and specifically on requirements to localize open source software. The paper also presents some concrete benefits for localizing FOSS and finally presents some challenges and recommendations on how to proceed towards FOSS localization.

2. Scope of Localization

Localization is defined in differing ways. The author would define it as enabling computing experience in linguistic culture of the user. Linguistic culture is not just limited to the language but how the language is used by the environment of the user. Thus, for Punjabi speakers in India, the computer display the language in Gurmukhi script and for Punjabi speakers in Pakistan, the same language would be displayed in Arabic script. This would entail defining and implementing many linguistic and computing standards and computer applications. This section overviews these standards and software.

2.1. Basic Linguistic and Computing Standards

One of the basic issues in local language computing is defining encoding standards. Early localization efforts were based on ad hoc encoding, initially because there were no standards available and eventually as a way to tie-in users with the applications these localization vendors were producing. However, with increasing demand on producing data which could be shared across applications, especially after the advent of the Internet, which necessitated cross-vendor and cross-platform sharing of data in multiple languages, first standardized code pages for individual languages and eventually Unicode emerged. Unicode is a single encoding standard designed to cover all languages of the world. However, the standard is still under definition and still does not cover all languages of the world completely, especially languages spoken by smaller populations. In addition, the encoding in the standard is a compromise between the technology and linguistic requirements, e.g. presentation forms for Arabic script. Unicode is now being increasingly used around the world.

In addition to standard encoding, keyboard layout is also a very important standard. Again, overtime, multiple standards have emerged for many languages. Keyboards or Keypads (for smaller devices) offer a standard way of inputting text in different languages. However, different languages with different sets of characters require different keyboard layouts to be defined. Computers pose more requirements than typewriters, so existing older versions of keyboard layouts need to be updated. The keyboard layout may be designed on basis of frequency analysis, ad hoc standards, or a combination of both. Vendors like Microsoft would only support a keyboard within their software if they have been standardized by respective populations and government(s). However, need to standardize keyboard layouts is not as critical for computers now as vendors like Microsoft are now providing keyboard layout configurators which users can use to define their own layouts.

Any significant data processing requires data sorting. Sorting is normally language and culture sensitive and must be defined clearly. This sorting definition is called collation sequence of characters for a language. Many languages of the world still need to define a crisp collation sequence, as the ones used for dictionaries are sometimes vague as they have been developed for human users, who do not require exact definitions like computers.

Equally important to provide the correct linguistic environment and experience is to define the locale for a language. This includes defining months of the year, days of the week, date and time formats, etc. Locales are culture dependent and thus locale for American English may be different from the English spoken in UK in some ways (e.g. date formats are different).

Even when local language computing is possible, sometimes it becomes difficulty to operate the ICT devices (like computers and PDAs etc.) because their interface is in English. Thus, the standard terminology used by these interfaces (e.g. vocabulary in the computer menus: File, Edit, Save, Save As, Exit, Insert etc.) must be translated into local languages in a coherent manner so that all devices start using same local language terminology. There are no international organization(s) which manage this in a holistic manner and must be done at national level.

2.2. Basic Localization Applications

The most basic applications for localization are those implementing input and output methods. Applications implementing input would define the keyboard layout as desired by the user and would map the keystrokes to the proper sequence of internal encoding, e.g. Unicode.

Output methods would enable computers to define, interpret and display a language on screen, printers and other output devices. This is done through definition of fonts. Initial fonts were based on English, which is a relatively simple writing style. However, many Asian scripts pose significant challenges and require complex fonts with significant amount of rules to generate the appropriate output. There are multiple font definition formats for different vendors like Microsoft, Apple, etc. Also the font standards vary depending on the ICTs, e.g., fonts for computers may not work for mobile devices, etc. Fonts and drivers for all these devices and formats need to be provided.

Sorting utilities implementing standardized collation algorithms for languages also need to be implemented. However, sometimes the text needs to be processed before it can be sorted, depending on the language and the encoding scheme being utilized. For example, for character Hamza in Urdu there are two Unicode points, which must be equated before they are sorted, as both must sort the same way. This process is called normalization.

2.3. Intermediate and Advanced Localization Software

Though not critical for providing basic language functionality, there are many language related software applications which enable a more rich and fulfilling local language computing experience to the users. This section introduces some of these applications.

A commonly used utility in desktop computing finds a text string and (optionally) replaces it with another one. However, searching and replacing text strings may involve differing algorithms across languages and thus this utility has to be defined separately for these languages. For example, English has small and capital letters which may or may not be searched separately, but languages using Arabic script do not have this concept of capitalization.

Lexicon represents a dictionary for a language for human and software application consumption. Humans users use language lexica for conventional use, e.g. in desktop publishing. However, these lexica are also used by other computer applications like Spell Checker, Machine Translation, etc. Lexica for these applications may become very advanced, extensive and abstract. Advance forms of lexica may also provide support for a thesaurus, which suggests synonyms and antonyms of words and is now normally used in desktop publishing to generate content on ICTs.

Different languages pose different challenges to technology. For example, some languages do not put a space between different words, e.g. Lao. Thus, it is not be possible for the computer program to determine where to break a sequence of characters at the end of a line. Humans can do it because they use a mental lexicon. Such (language dependent) applications come within the scope of Natural Language Processor.

Spell Checker is used to check spelling in desktop publishing applications. It normally requires a lexicon and implements advanced searching algorithms. A good spell checker would give minimal possible suggestions ensuring that the word the user intended is among the suggestions provided.

Grammar Checker is used to check grammar in desktop publishing applications. It normally requires a lexicon and encodes the grammar of the language. Precise grammar definition for the language is a prerequisite for developing a good grammar checker.

Text to Speech system takes regular text in a language from a file or keyboard as input and converts it to speech in that language using linguistic knowledge and digital signal processing techniques. TTS system is especially useful tool to access content for vision or speech related disabilities and for illiterate people.

Speech Recognition system is basically a dictation system, which can enable computer to type what the user is speaking. Speech recognition systems are mostly speaker dependent (for higher accuracy) and therefore must be trained before they can be used. They are very good tools for content creation, especially for illiterate populations.

Machine Translation system translates text from source language to target language automatically using grammars of both languages involved. Currently, though one could get a grammatically correct translation, meaningful (semantically correct) translations are hard to get automatically and human quality assurance is normally required. However, research and development in this area is improving the quality of MT. Even rudimentary MTs can bring the enormous English and Spanish content already available on the net to local populations, the former being the source languages and the latter being the target languages.

Optical Character Recognition system converts scanned documents into text automatically. Thus, instead of typing up records and books, one could just scan and process it through OCR to create online content very quickly and effectively. OCR normally requires advanced image processing and is potentially a great tool for content creation. An advanced form of OCR (though sometimes involving different processing techniques) is the Handwriting Recognition system that can also recognize human writing, which has much more variability.

2.4. Other Localization Support

In addition to providing standards and software for localization, training to use the software is an integral part of the localization process. Minimally, manuals need to be translated in local languages for using computers. However, in most cases, where users are new to the computing environment, more formal training is needed.

2.5. Overview of Process for Localizing FOSS

Localization of FOSS would mean enabling Linux environment in local languages. This would involve the standardization and application development process detailed in Section 2 above. Unicode standard for encoding, keyboard layout, collation sequence, locale and display local language terminology would all need to be implemented in basic Linux distribution. For font display the basic utilities already provided would need to be enhanced, e.g. XFree and Pango rendering engines. Mozzilla application would also need to be enhanced to enable access to email and internet. Translations of terminology would need to be embedded in the appropriate (.PO) files distributed within Linux. In addition, to provide even limited word-processing capabilities in local languages, GNOME and or KDE environments would also need to be converted to local languages. Finally, all the help files and user manuals would need to be translated. For more advanced desktop publishing, applications like Open Office would need to be enhanced to support the language(s) being considered. Finally, all this would need to be packaged into a Linux distribution for circulation. This requires a significant amount of effort as there is very limited documentation and support available for the complete process at any one place.

3. Benefits of Localization of FOSS

- As pointed out a very small minority of Asians access the Internet. Providing support to ICTs in local languages would be instrumental in providing access to the 95% population of Asia, still not using the Internet. This would present an unprecedented potential for development of urban and rural populations of Asia.

- In addition to being in local languages, this software will also be free. Thus, is would be affordable to the vast developing majority of Asia.

- As FOSS is not very resource-hungry, it can be effectively deployed on older and cheaper computers (e.g. the initiative by NECTEC, Thailand to distribute Thai Linux with PCs for USD 200). This also increases the affordability of the computing and thus access to information for the underdeveloped populations.

- Many underdeveloped countries are too small and are not as lucrative an option for private vendors to provide local language support, e.g. Bhutan, Cambodia, Laos, Sri Lanka are still looking forward to complete support of their languages by Microsoft. Linux provides an alternative to develop national solutions for such languages.

- About 4% or total language of the world (about 6000) are spoken by 96% of world population and rest 96% of the languages are spoken by only 4% of the world's population. Some of these languages are endangered due to decreasing populations of people who speak these languages. Providing FOSS could provide one way of saving these languages form extinction. Again, commercial vendors may not support such languages for a very long time.

4. Challenges and Recommendations for Developing Localized FOSS

Though localizing FOSS is a very attractive option for developing countries, it is by no means an easy option. First and foremost, open source software is a movement which is largely based on contributions of geographically scattered volunteers, as it presents no commercial incentives. This distributed nature of open source has made it possible for this technology to become formidable. However, the development is not planned and does not follow strict timelines. National governments should take up this responsibility, by formalizing Linux development in their national languages by formally adopting existing groups or developing newer ones if none exist. Only if some funding is given to this cause, a country can achieve planned and targeted development of FOSS.

Though some countries can manage to develop their indigenous solutions to support their languages through ICTs, many countries do not have appropriate human resource capability. One of the main reasons is that local language computing development requires cross-disciplinary research and training. Computational linguistic applications require knowledge of linguistics and computer sciences. Speech processing applications require knowledge of engineering and speech science. These multidisciplinary fields are relatively new and more challenging as compared to the more traditional fields. Special training is therefore required in many of these countries to enable human resources to develop their language solutions. This unavailability of resources is even more pronounced for localizing open source software. With very limited structured help available on the process, which is scattered over the internet, it is very hard to develop the available human resource. It is also not possible for an expert of localization of FOSS in one language, to develop localization in a different language due to language specific complexities. Thus, there is real and urgent need for collaborative work in this area.

Even where resources are available, political will or awareness to invest into local language computing and FOSS in the "power circles" is missing. Thus, many ICT policies do not address these issues. Where the policies do address this issue, the policy makers do not fully understand the depth and breadth of the problems, allocating inadequate funds and having unreasonable expectations or misplaced priorities. For example, even to-date cutting edge Machine Translation solutions only perform to the accuracy of 40-60% for many European language pairs (e.g. English to German translation is still about 40% at best). However, when governments sanction such projects they may expect miracles, within limited budget and time. Thus, a comprehensive look at local language computing of FOSS from the viewpoint of ICT policy is also required.

Though many countries are struggling to provide support in ICTs for their languages, there is no regional collaborative effort which can bring together these efforts at a common platform to prevent everybody from re-inventing the wheel. The author's personal experience has shown that similar technology development is required by most of the countries and a lot of effort and time can be saved if experiences can be shared across linguistic and political boundaries.

By developing a detailed How-to guide for localizing Linux, organizing regional trainings and sponsoring regional support networks, many of the challenges posed above can be addressed more effectively. It would also be very useful to develop software tools which can automate the localization process for new languages as much as possible. The usage can also be enhanced by providing user-level trainings, once FOSS is localized for a particular language.

Both national and regional will needs to be developed to address the challenges posed by FOSS localization to eventually reap the immense benefits it promises.