Enabling Indic Support in Library Information Systems : An Opensource Localizer's Perspective

Indranil Das Gupta Najmun Nessa

Abstract

This article looks into the unique nature of challenges and opportunities facing the Free & Opensource (F/OSS) based software localizers' community when it comes to enabling support for Unicode-based Indic Scripts in the domain of Library & Information Science (LIS). It describes the early background of Indian language support in LIS domain in terms of technology used, and moves into the present-day scenario of Unicode & Open standard based method of universal archival and access to information repositories that modern libraries represent with their multi-media capabilities. Unicode addresses many of the problems that had plagued earlier systems which had little or no capabilities in terms of universal accessibility, it also brings its own set of problems that demand solutions – e.g. the issue of collation sequences which assume significance when looked at from the perspective of indexed search capabilities in library software. While Opensource provides an open, pro-active, collaborative platform for rapid development, it still has to answer for issues like availability of extensive Opentype fonts, collation sequences, less-than desired quality of rendering by Indic script layout engines, as well as varying levels of maturity of software components that make up the technology stack on which Indic Support enabled Library Information Systems can and are being developed. The authors will try to seek answers to these practical questions by looking into their localization experiences with Koha – the world's first Opensource library software into Bengali (this work is being followed by Hindi localization). Inputs will also include the experiences of the team from ISI, Kolkata which is working on localizing Greenstone Digital Library (GSDL) into Bengali. The article will draw upon the experiences of F/OSS Indic Localizers' community to see whether cross-pollination of ideas can lead us towards the goal of bridging the Digital Divide.

Keywords : Indic Scripts, Unicode, Localization, Library Automation Software.

0. Information Divide & the Emerging Role of F/OSS[1] in ICT4D

In the present day world, information technology (IT) is a key part of the infrastructure development. ICT (Information & Communication technologies) penetration is being measured as part of development indices. Access to information and the information technology has emerged as a key to the development in any sector – be that education, access to health-care, access to markets for rural produce, to overseas trade, entrepreneurial development, public & private investment, and even the governance of a country.

Economic disparities separate the developed nations from the rest. As a result, the developing nations are lagging behind in adoption of IT. This is further aggravated by factors like poor literacy rates, multi-lingual societies with little or no comprehension of English (which is the *de-facto* language of IT). All these factors has given the English dictionary a new word – The Digital Divide.

Today the Digital Divide has emerged as one of the primary obstacles to development. Within countries like India, Brazil & China, it has assumed far greater complicacy because in these countries there has emerged the domestic Digital Divide. People within the country who have access to the latest in information technology while the majority of the population does not have even the most basic mode of access to IT.

The emergence of Free/Opensource software as a global phenomena has resulted in redrawing the ICT4D strategy maps. All across the developing world, the access to the underlying technology (via the programs' source code) and the license to add, modify and improve – either at will or driven by a need to address specific requirements, has opened windows of opportunities of hitherto unparalleled dimensions and importance[2].

1. Relevance of these Developments in Library Information Systems

Through the ages, libraries have been the cornerstone of recorded information collectively available to any society. The emergence of the Internet as an affordable, instant, global, digital communication medium has perhaps brought about the most significant change since Gutenberg's invention of the printing press as to how libraries acquire, disseminate and manage information.

Media changes notwithstanding, libraries have acquired greater significance in the modern knowledge-based societies with their well-defined classification, cataloging services. In developing countries they have a far greater role to play in bridging the Information/Digital Divide.

In India, the demand for delivering timely, multi-lingual, multi-media based digital content has emerged as the need of the day. Special libraries services have sprung up to cater to these needs. The much talked-about Vidyanidhi[3] Project is one such key example.

The reason we chose to address Library Information Systems as against taking up only Integrated Library Management Systems (ILSes, similar to LibSys etc) or Digital Library Software (GSDL and DSpace et. al.) is quite simple. The problems solved, problems pending and the lessons learnt in course of developing Indic-enabled F/OSS solutions are equally applicable across the range.

Being native speakers of the Bangla (BN_IN) language and being actively involved in BN Localization (L10N) efforts as part of AnkurBangla Project, IndLinux Consortium, and L2C2 Initiatives, examples used will heavily depend on our experiences with Bengali. However, most of the examples, except for the very specific ones, apply across most Indian languages.

Our work on localizing Koha or for that matter others' initiatives on F/OSS solutions like the Greenstone Digital Library[4] Software (GSDL) or DSpace[5] couldn't have happened without the Indic Language support being in place on the Free & Opensource platform. We believe that it is essential to share the basic know-how of localization on F/OSS with the Library & Information Science community, now that F/OSS Indic support have become mature enough to provide support to localization of 3rd party software.

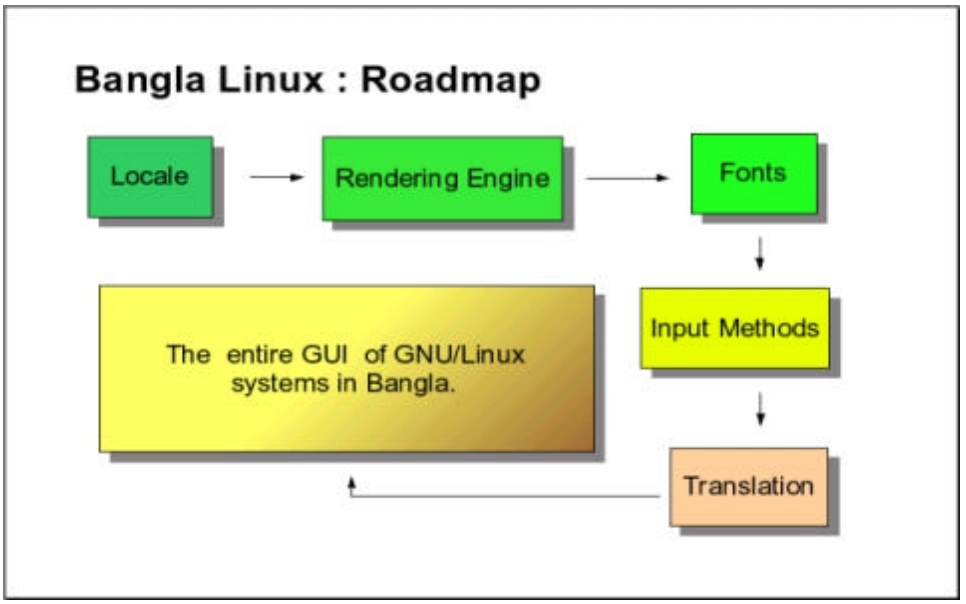
In the ensuing discussion, references to F/OSS will primarily focus of OSS software working on the GNU/Linux[6] Operating System which has the most mature Indic support among other Free OSes.

2. The F/OSS based Indic Localization Roadmap

F/OSS-based development models have always been collaborative from a pluralistic sense. Since its early days it was *assumed* (unlike in the case of a lot of proprietary systems) that the software being developed *would* be used by non-english speaking users. This is not surprising since developers across the globe would collaborate on projects using the Internet as the ultimate project management platform.

So, when F/OSS based Indic Localization (L10N) initiatives got off the ground, the basic software engineering framework was already in place. It goes without saying that this framework did have its shortcomings which has since been addressed. With the framework in place, Indic Localizers could focus on creating the basic artefacts required to deliver a L10N-ised platform to the end-users.

These basic artefacts included – Unicode support, creation/correction of Locale Data[7], correcting/modifying rendering & layout engine programming code, creation of OpenType Fonts, creating Input Methods for text entry and finally User-Interface (UI) translation. Below is a slide from an AnkurBangla presentation depicting the basic components needed to deliver the Bangla GUI Interface on Linux. This applies in case of all Indic Languages.



NB : The above slide omits a major component in the localization technology stack – the *collation sequences*. We shall come back to that in due course.

We shall take a closer look at some of these components, as the issues to be presented applies equally in case of Library Information Systems as in any other application domain based on F/OSS platform as we found in course of our work.

3. Indian Standard Code for Information Interchange (ISCII) – The Past

During the late-80's and early 90's C-DAC[8] (Centre for Development of Advanced Computing), then under the Department of Electronics, Govt of India, created a standard called ISCII (Indian Standard Code for Information Interchange) for use of Indian Languages on Computers.

ISCII uses a 8-bit encoding that uses escape sequences to announce the particular Indic script represented by a following coded character sequence. The ISCII document is IS13194:1991, available from the BIS offices.

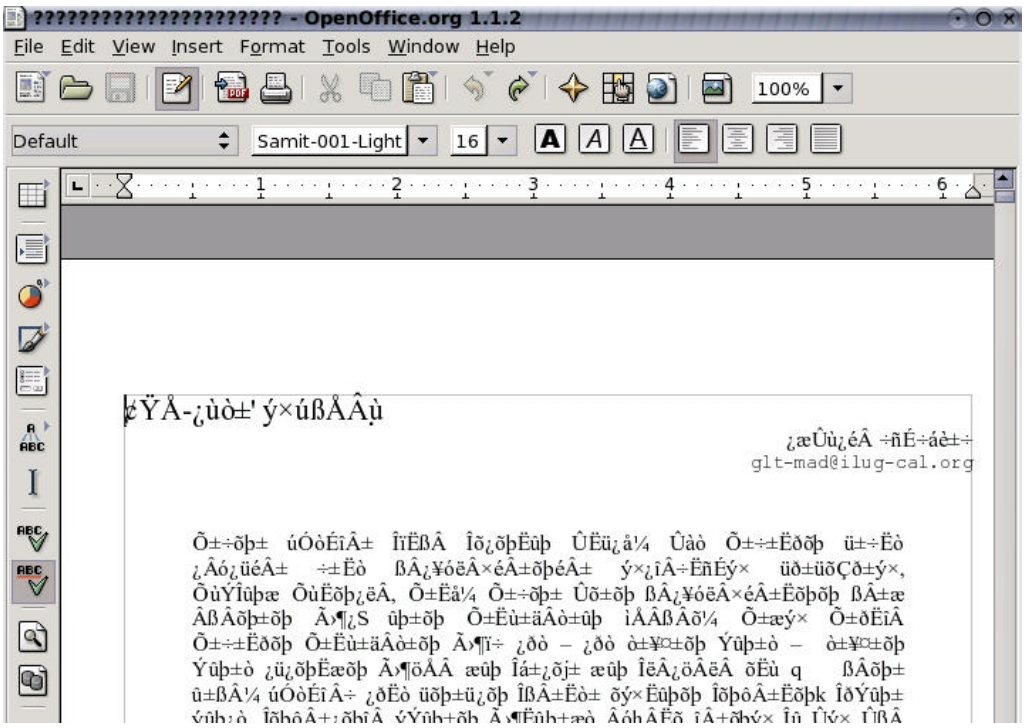
Alongside ISCII, other proprietary Indian language solutions did exist prior to Unicode. The most criticized aspect of these developments was the proliferation of encoded fonts using closed, proprietary formats. As a result, most of these solutions didn't (or rather couldn't) exchange data between themselves or other software. Vendors who created these software did this on purpose to ensure lock-in of the users to their specific software.

4. Unicode – New Challenges, New Possibilities

Unicode – a plain text standard, which is an idea of simplicity, promises to change all that for the future. All major operating systems (Windows, Mac OSX, Linux etc.) today support Unicode as a data format. Most are even beginning to support Unicode at GUI levels. As a result, on Unicode enabled platforms, it now possible to copy a piece of text written in Unicoded Hindi, paste it into a web-page that you are designing or store it AS-IS into a RDBMS like Oracle, Sybase or PostgreSQL.

Fundamentally, computers just deal with numbers. They store letters and other characters by assigning a number for each one. Before Unicode was invented, there were hundreds of different encoding systems for assigning these numbers. These encoding systems also conflict with one another. That is, two encodings can use the same number for two *different* characters, or use different numbers for the *same* character. Any given computer (especially servers) needs to support many different encodings; yet whenever data is passed between different encodings or platforms, that data always runs the risk of corruption.

Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language. The emergence of the Unicode Standard, and the availability of tools supporting it, are among the most significant recent global software technology trends.



A document written in Bangla in old TTF Font using custom encoding

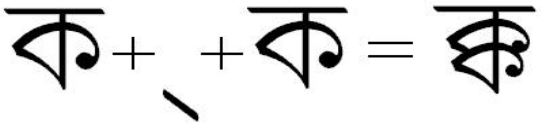
While Unicode seems like the likely answer for long-term adoption of Indic Language software, the immediate questions it throws up are also quite a few. As a standard it is still evolving. One needs to use OpenType fonts with Unicode which aren't that many in number as yet compared to earlier, proprietary solutions. And of course, one needs to look at the exist corpus of data stored in earlier encodings.

ISCII to Unicode converters are fairly well-available in the Opensource. But problems are often faced with converting documents that use TrueType fonts based on closed, proprietary encoding schemes.

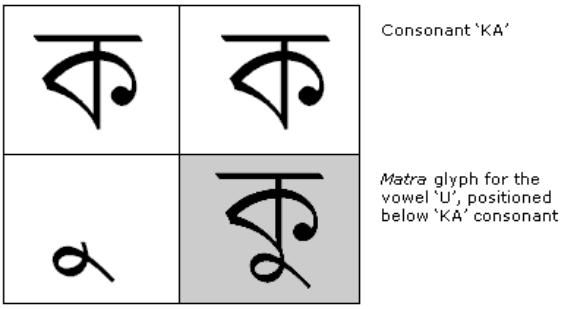
5. Open Type Fonts

The OpenType specification says – "In OpenType all the information controlling the substitution and relative positioning of glyphs during glyph processing is contained within the font itself. This information is defined in OpenType Layout (OTL) features that are, in turn, associated with specific scripts and language systems. Placing control of glyph substitution and positioning directly in the font puts a great deal of responsibility for the success of complicated glyph processing on the shoulders of type designers and font developers, but since the work involves making decisions about the appearance of text, this is the correct place for the responsibility to land. OpenType font developers enjoy a great deal of freedom in defining what features are suitable to a particular typeface design, but they remain dependent on application support to make those features accessible to users."

To understand how they actually work mention needs to be made of "two internal tables need to be introduced now. These are the GSUB and GPOS tables that contain instructions for, respectively, glyph substitution and glyph positioning. Glyph substitution involves replacing one or more glyphs with one or more different glyphs representing the same text string The backing string of Unicode characters is not changed, only the visual representation. These substitutions may be required (as part of script rendering), recommended as default behavior, or activated at the discretion of the user; they may also be contextual, active only when preceded or followed by a certain glyph or sequence of glyphs, or contextually chained so that one substitution affects another."



Example of use of GSUB table for glyph substitution



Example of use of GPOS data for glyph positioning

The list[9] below describes some to the GPLed Bangla OTF fonts. As is evident, the number of glyphs supported by each font varies often by a wide margin.

* Akaash – 409 characters (642 glyphs) in version 0.75
  Ranges: Basic Latin; Latin-1 Supplement; Latin Extended-A; Bengali
  OpenType layout tables: Bengali, Devanagari, Latin
  Family: Serif
  Styles: Normal
  Availability: Free download from The Free Bangla Fonts Project[10]

* Likhan – 286 characters (746 glyphs) in version 001.100
  Ranges: Basic Latin; Bengali
  OpenType layout tables: Bengali
  Family: Sans-serif
  Styles: Medium
  Availability: Free download from The Free Bangla Fonts Project

* Mitra Mono – 250 characters (324 glyphs) in version 0.70
  Ranges: Basic Latin; Latin-1 Supplement; Bengali
  OpenType layout tables: Bengali
  Family: Monospace (but Latin characters are not fixed width)
  Styles: Regular
  Availability: Free download from The Free Bangla Fonts Project

* Mukti – 197 characters (562 glyphs) in version 0.92
  Ranges: Basic Latin; Bengali
  OpenType layout tables: Bengali
  Family: Serif
  Styles: Regular, Bold
  Availability: Free download from The Free Bangla Fonts Project

* Mukti Narrow – 197 characters (562 glyphs) in version 0.92
  Ranges: Basic Latin; Bengali
  OpenType layout tables: Bengali
  Family: Sans-serif
  Styles: Regular, Bold
  Availability: Free download from The Free Bangla Fonts Project

* UniBangla – 184 characters (329 glyphs) in version 1.0

  Ranges: Basic Latin (non-alphanumeric); Bengali

  OpenType layout tables: Bengali, Devanagari, Latin

  Family: Sans-serif

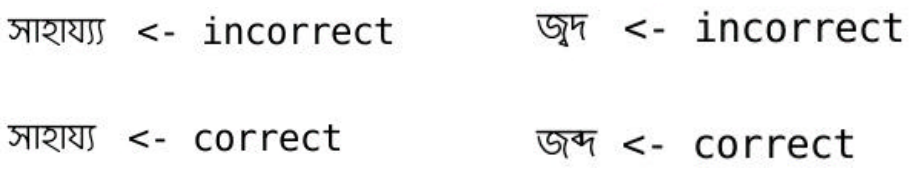
  Styles: Normal

  Availability: Free download from BanglaLinux[11]

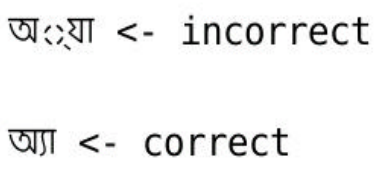
6. Engineering Pango – The GTK/GNOME Rendering Engine

Among the F/OSS Desktop Environments – GNOME[12] has witnessed the maximum work done in Indic L10N domain. The reason was simple. GNOME was the first one to provide support for Indic Scripts through its rendering engine – Pango. It is Pango which takes the Unicoded data, identifies the correct OpenType[13] font (assuming that one is installed), applies text layout processing rules that a language may need (e.g. sanjuktaakshars or conjunct clusters) and render it on-screen.

Pango has seen some major improvements in the last 18 months. Earlier releases had rendering issues affecting the Bengali *ya-phala, ba-phala* marks, handling of ZWJ/ZWNJ control characters etc[14]. These were addressed by members of AnkurBangla Project. The images below describe the ante & post situations.



The ya-phala issue #1 *The ba-phala issue*



The ya-phala issue #2

Later on other issues like the INIT feature[15] which is essential for the Bengali script were taken up and corrected by AnkurBangla developers. The image on the left hand side shows the INIT feature properly, whereas on the right side is the older, incorrect rendering of the script.

7. Collation Sequences

In simple words, collation sequences define the sorting order in any given language locale. A simple, implicit way, is through code-point ordering where the order is based on the numerical ordering of code points e.g. in ASCII A = 65, B = 66, C = 67 etc.

The reason why it assumes such significance in Indic Unicode-based systems is that several Indic language share the same script due to commonality in their source of origin (Hindi, Marathi, Sanksrit, Konkani), and others have scripts that are very similar (Tamil-Malayalam, Kannada-Telugu).

Unicode charts assigned to Indic scripts make no distinction between languages. Therefore, some charts use the same code chart for the following languages:

1. Devanagari: Hindi, Marathi, Sanskrit, Konkani, Nepali
2. Bengali: Bengali, Assamese, Manipuri
3. Arabic: Urdu, Kashmiri, Sindhi

The ISCII-88 standard (Indian language block of Unicode Standard is based on ISCII-88[16]) was based on phonetic commonality rather than correct sorting sequence. This distorted some traditional sorting conventions, and developers should not interpret the character sequence to be the same as their collation sequence. For example, though Hindi and Marathi use the Devanagari Unicode charts, the Hindi sorting sequence is not the same as Marathi. Similar situations exist in case of Assamese and Bengali which share the same script but a different ordering sequence. This requires that sorting be tailored to languages rather than scripts.

In multi-lingual Indic-enabled Library Information systems, the collation data is used/needed by sort/ search routines, and is therefore vital for their efficient operation. Collation data is defined in LC_COLLATE category in locale definition. A default approach (as currently done for Indic locales) is to copy iso14651_t1 table like

```
LC_COLLATE
% Copy the template from ISO/IEC 14651
copy “iso14651_t1”
END LC_COLLATE
```

The above table (stored in /usr/share/i18n/locales/iso14651_t1) doesn’t contain any data for Indic script ranges. So Indic sorting defaults to code points based, as in the Unicode charts. This behavior is defined by the Unicode Collation Algorithm (UCA), providing a default sort order that may be used only when no additional information is available. It can be found in the Unicode Technical Standard #10[17] document.

Code-point based sorting is good enough for simple scripts like Latin, most European scripts where the number of characters is less and do not generally combine with other characters (like *sanjuktaakshars* in Indic Languages). The disadvantage with code point sorting is its fixed forever, and if the encoded script (say Devanagari) were to be used with multiple languages (say Hindi, Nepali & Marathi), having different rules for sorting then it becomes difficult to accommodate them. Since many scripts are common across region/languages its imperative that collation sequence is independent of encoding.

8. Localizing Koha

Having covered the technical background on which the actual work of localization was/is being done, its time we turn towards Koha.[18] About this award-winning software, Joshua Ferraro – a leading Koha developer, says on his website[19]:

"Koha was built using Perl scripting language[20], MySQL Relational Database Management System[21], and Apache[22] Web Server, running on the GNU/Linux Operating System[23]; however, it has been ported to other operating systems (including Windows) and should be compatible with any system running an SQL database, a web server, and Perl."

Both of us had been engaged in the setting up of technical infrastructure of the library at West Bengal University of Technology[24]. During the first phase, it involved implementing a fully F/OSS-based ILS system using Koha which was completely browser-based. And during the second phase, we were engaged in setting up a digital library by extending Koha's capabilities. However, *that* is a different case study altogether.

It was during a discuss with librarians from another organization who were commenting about lack of Indic support in their present Library management system that the idea of localizing Koha came up.

9. The Requirements for Localization

To localize a browser-based software like Koha, a localizer has to address the following potential problem areas:

1. Making sure that the relational database management backend supports Unicode.
2. The server-side scripting engine (PERL in this case) must support regular expressions and strings with Unicode embedded.
3. The web server (Apache 2.0.48) must be capable of handling UTF-8 as a native CHARSET (character set).
4. The browser used on the client systems (Mozilla for us) must support Unicode and have complete rendering support for the script in question (Bengali) into which Koha is to be localized.

10. Creating the Localized Computation Infrastructure

With Koha being deploying on the Fedora Core 2[25] Linux platform, it was essential *not to install* the older version of MySQL database server which came with it. The latest 4.0.x range of MySQL server software which is Unicode compliant, was downloaded, compiled and installed on server.

Using the browser-based MySQL admin tool called phpmyadmin, the database was tested if it was handling Indic language strings properly. INSERT, UPDATE and SELECT SQL statements were all used in different permutations and combinations of Bengali, Hindi & Urdu data strings, to find out the stability of the platform.

Apache webserver was the next in line and we needed to make sure that the list of AddCharset directives in the apache configuration file was listing UTF-8 among others. This was to make sure that the webserver could serve out content (in this case Koha) using UTF-8 character set encoding.

We tested this by uploading a webpage with its charset attribute set to UTF-8 and then trapping the http headers to to see if we were being served the right charset content.

Once these tests proved OK and the system stable enough, it was time to pull in the latest Koha source code from its CVS server on sourceforge[26].

11. Translating Koha

After downloading the latest sources, we tried to follow the instructions provided for people interested in translating Koha into other languages. There are essentially two areas that one has to translate – the administrative intranet interface; and the Online Public Access Catalog (OPAC) interface that is visible to the general public once the Koha server is online. Presently it totals about ~4000 strings in all. Of course, this number doesn't include the online, context sensitive help documentation.

It turned out that the translation help documentation bundled with the sources was out-of-date and didn't give the desired output. Being opensource software, a quick look at the code inside brought about the following conclusion that instead of tmpl_process.pl in the misc/translator directory, one should use the new tmpl_process3.pl script from the same directory.

The tmpl_process3.pl script created the all-important POT (Portable Object Template) file which is the backbone of any translation effort on GNU/Linux. The script internally calls upon the GNU Gettext[27] Internationalization (i18n) library using PERL modules.

The next step was to rename the .pot files into the target language .po files – in this case, it was renamed from default_intranet.pot to default_intranet_bn_IN.po and css_opac.pot to css_opac_bn_IN.po following standard L10N conventions and start the actual translation. Once the translation is done the script is again used to generate the HTML templates in the desired language (i.e. one presently translated to).

12. Translation Methodology

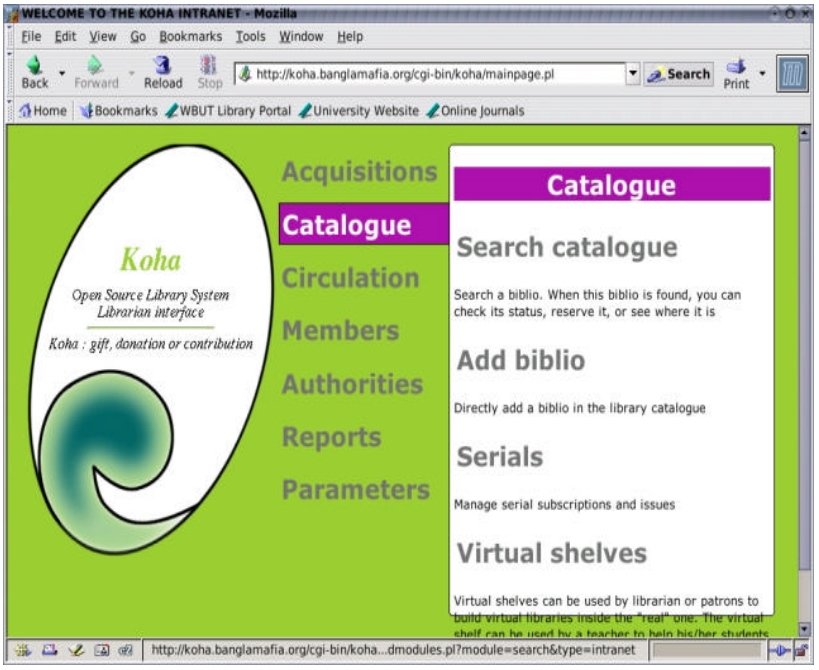
Our experiences during work on the AnkurBangla Project had equipped us fairly well in terms of selecting terminology or creating new ones wherever needed. But in case of Koha, a different route was taken. With the library movement having taken deep roots in West Bengal, there has been numerous efforts to created Library and Information Science Terminology in Bengali.

Using the reference library at Bengal Library Association (Bangiyo Granthagar Parishad) a a cross-reference glossory of terms from the domain was created and this was used extensively during the translation. Where terms from IT were encountered, the glossaries from AnkurBangla Project were used.

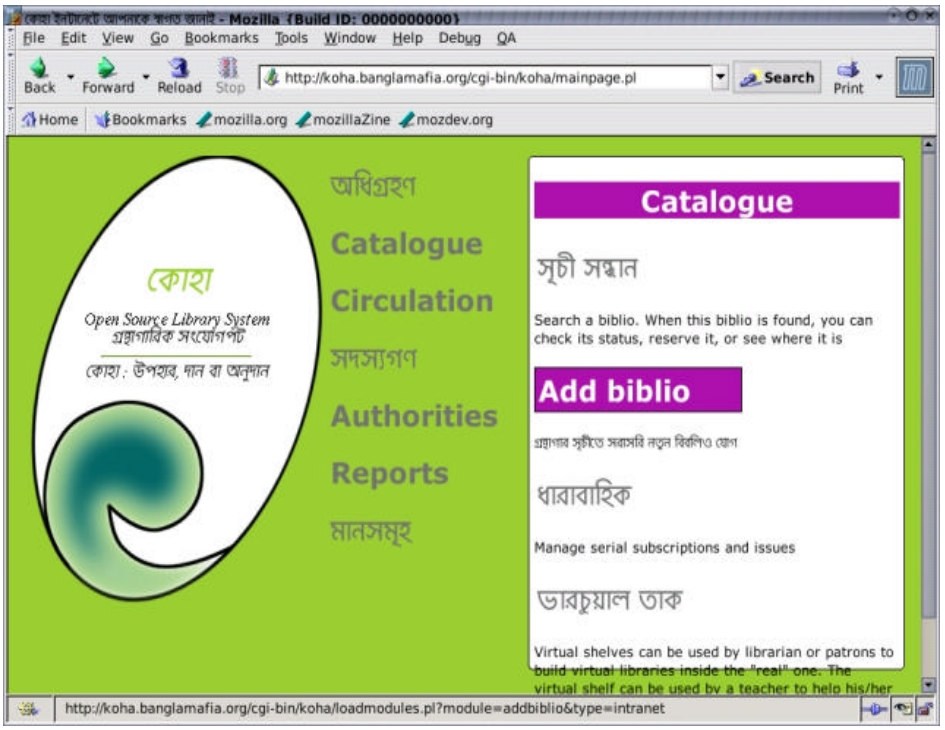
Use of existing terminological referencing was done to ensure that the localized interface of Koha along with it OPAC would be readily acceptable and meaningful to people accustomed to the Bengali terms.

13. Testing the Localized Koha

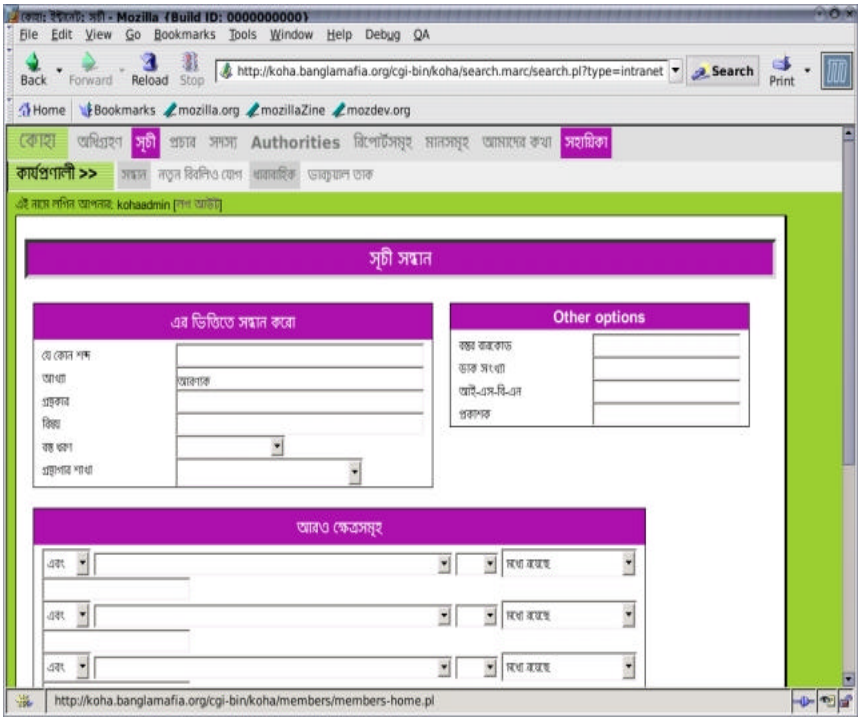
As the saying goes, the proof of the pudding is in eating it, so it was no different for a software which was being localized. Aside from translation, changes to the HTML templating code also had to be made as the existing code was forcing the CHARSET attribute to be iso-8859-1 and not UTF-8. This was causing loss of data in-transit as well as junked on-screen rendering.



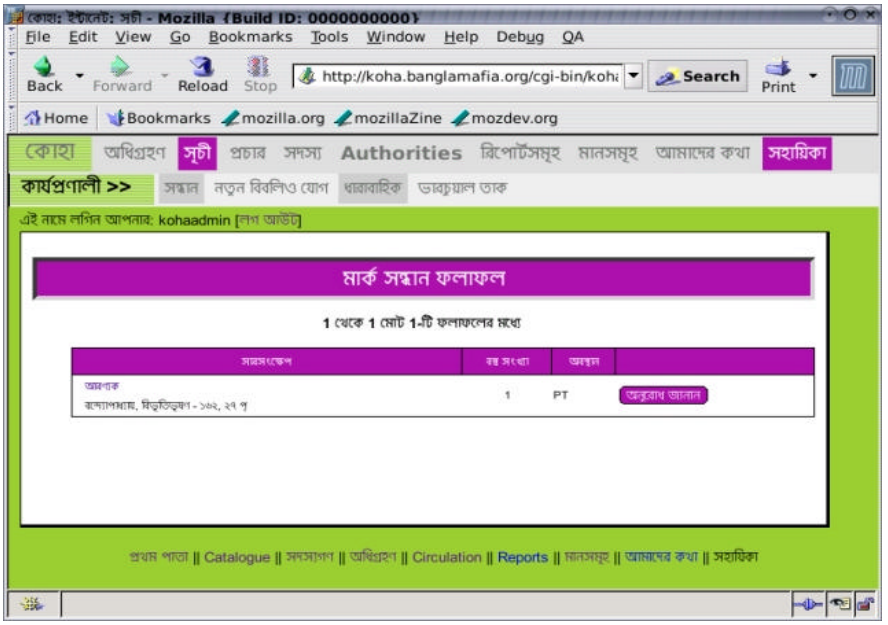
The default english interface of Koha's admin UI



Same interface but in Bengali (partially translated)



The Catalog Search in Bengali for a Title in Bengali



Search Result in Bengali (1 item found)

14. Status of The Project

The localization of Koha is presently in an advanced state. It is expected that it will be finished within the first week of January 2005, which is also when Koha 2.2 (the next version) is scheduled for release. It is expected that with the release of Koha 2.2, Bengali would be the first South Asia language to be supported in Koha. This project is completely non-funded and work done on it is on a volunteer basis by us in our spare time.

During the International Summit Conference held at New Delhi during 8 – 10 December 2004, dialog was initiated with persons from the SARAI[28] unit of CSDS (Center for Studies in Developing Societies), New Delhi, for beginning work in the Hindi & Urdu Localization of Koha from February 2005 onwards.

15. Other Similar Projects

In recent times there has been other efforts in similar directions. The team of Prasenjit Majumdar, Dr. Mandar Mitra & Rajdeep Mukherjee – all from ISI, Kolkata have been working on localizing the opensource UNESCO-funded Greenstone Digital Library (GSDL) Project. They have been using Windows XP to carry on the work of translations. Their work too is expected to be completed by January 2005. The Vidyanidhi project too has been investing in localizing DSpace, in order to acquire support for languages such as Hindi and Kannada.

16. Conclusion

Serious short-comings exist like the lack of collation sequence data on GNU/Linux OS platform. This affects the quality and speed of database searching which is presently based only on code-point ordering sequence. This problem is likely to get addressed with the next release of CLDR data (*See earlier reference to Locale data*).

Also, the sorting & searching algorithms, Natural Language Processing (NLP) which work well on linear Latin based scripts do not work quite the same way when applied to our complex, re-ordered text layouts. Research and development needs to be undertaken in the F/OSS domain to carry forward the task of Indic support in the areas of NLP.

Inspite of all these issues, today the ground for Indic localization of library information systems based on Free / Opensource platforms is well-prepared for the march ahead. It is a march towards a future based on standards-driven, internationally compatible systems offering equitable access to information to all who are in need of information in a language of their own.

17. References

1 Free & Opensource Software

2 For example, the Extremadura province in Spain once had the lowest PC:student ratio in schools in Europe, since the initiation of Project LinEx (mass adoption program of Linux Operating System PCs in education sector) it now among the highest. [Ref : http://www.linuxjournal.com/article/7908]

3 The Vidyanidhi Project [http://www.vidyanidhi.org.in]

4 Greenstone Digital Library Software [http://www.greenstone.org/]

5 DSpace Federation [http://www.dspace.org/]

6 www.kernel.org

7 Common Locale Data Repository (CLDR) Project [http://www.unicode.org/cldr/]
8 http://www.cdacindia.com/index.asp
9 http://www.alanwood.net/unicode/fonts.html#bengali
10 Free Bangla Fonts Project [http://www.nongnu.org/freebangfont/]
11 http://www.sourceforge.net/projects/banglalinux/
12 The GNOME Project [http://www.gnome.org/]
13 Microsoft Typography – OpenType Specification [http://www.microsoft.com/OpenType/OTSpec/]
14 Bugs in the Bengali rendering system of Pango [http://bugzilla.gnome.org/show_bug.cgi?id=113551]
15 Bengali Opentype Specification [http://www.microsoft.com/typography/otfntdev/bengalot/features.htm]
16 http://www.cdacindia.com/html/gist/standard/unicode.asp
17 Unicode Technical Standard #10 : Unicode Collation Algorithm [http://www.unicode.org/reports/tr10/]
18 http://www.koha.org
19 http://kados.org/LibraryScience/koha_at_a_glance.html
20 http://perl.org
21 http://mysql.org
22 http://www.apache.org
23 http://kernel.org
24 West Bengal University of Technology [http://www.wbut.net]
25 The Fedora Project [http://fedora.redhat.com]
26 http://sourceforge.net
27 The GNU gettext project [http://www.gnu.org/software/gettext/]
28 SARAI [http://sarai.net]

About Authors

Mr. Indranil Das Gupta has been an active user and evangelist for Free & Open Source software for the past several years. Aside from his vocation as consultant helping in managing the adoption and migration to Free & Opensource technologies, he has been active in the area of Localization of Free/Opensource Software in Indian Languages. As part of the IndLinux Group (www.indlinux.org), he is currently trying to create a model for productization of Indic F/OSS initiatives for mass-scale adoption using the L2C2 framework. Heis presently based in Kolkata.
E-mail : indradg@l2c2.org

Ms. Najmun Nessa is presently working at the West Bengal University of Technology, Kolkata (www.wbut.net) as an Asst. Librarian. She holds a Masters Degree in Library and Information Science from Jadavpur University, Kolkata. Along with Das Gupta she has been setting up a completely Opensource based Integrated Library Management System using Koha alongside establishing a Digital Library at this fledgling University. A founder-member of Indian Koha Interest Group, and she is keenly interested in Indic Language Cataloguing in digital formats. As a step towards that direction she has worked on localizing Koha version 2.2 to Bengali in collaboration with Das Gupta.
E-mail : najmunnessa@yahoo.com