International
Open
Source
Network

Free/Open Source Software

Guide to Localisation

Sasikumar M, Aparna R
Naveen K and Rajendra Prasad M

Asia-Pacific Development Information Programme
e-Primers on Free/Open Source Software

Centre for Development of Advanced Computing
CDAC Kharghar
Mumbai, Maharashtra 400614
http://www.ncst.ernet.in/

Published March 2005

Table of Contents

List of Tables

List of Figures

Chapter 1. How to Use This Guide?

1.1. Background

Read if: Well, there is no "if" here. You must read this chapter, rather small, first.

You are part of a community who speaks and uses a not-so internationally popular language, let us call it hypolan[1] for ease of reference. To help your community to benefit from the ICT revolution sweeping across the globe, you would like to enable computers to communicate in hypolan. Knowing the Open Source Software -- so well documented in the literature --, you opt to base your work on Linux and associated software suite. Where does one start? What are the steps involved? What resources do you need? What are the requirements on these resources? What tools and information outlets are available to help you with the various parts of this task? We all know there is a large and generous community of developers and evangelists across the world. But identifying the right forum for a given problem is a non-trivial challenge. This how-to guide is for YOU.

This is not an exhaustive treatise on localisation, but more a practitioner's guide to localisation. We will discuss some of the important concepts one must know when embarking on a task such as localisation, partly to communicate freely with other colleagues and community, and partly to get a better understanding of the process involved. We will also give links and references to relevant resources for those who want to explore any of the issues in detail.

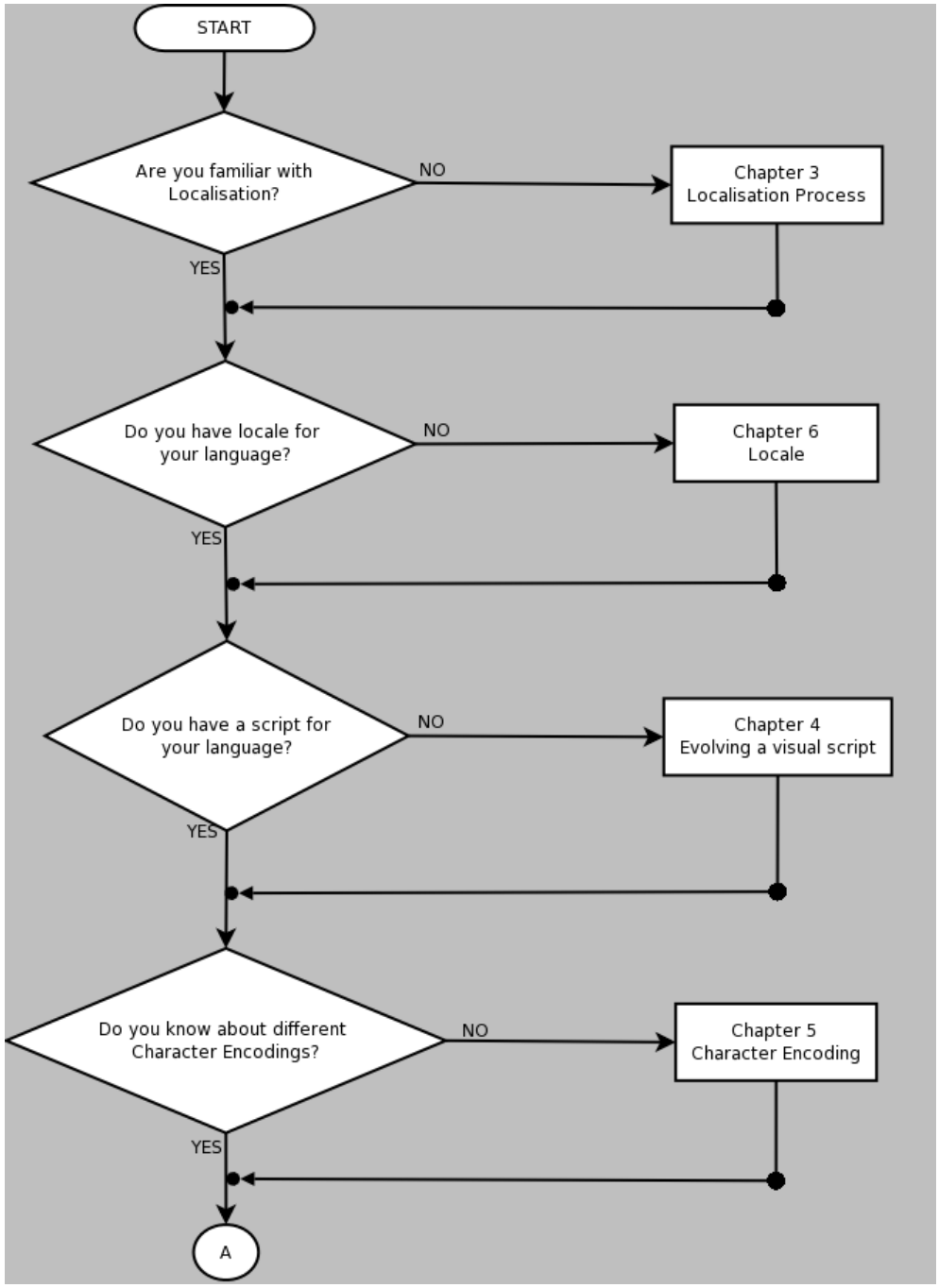
1.2. So Where do I Start?

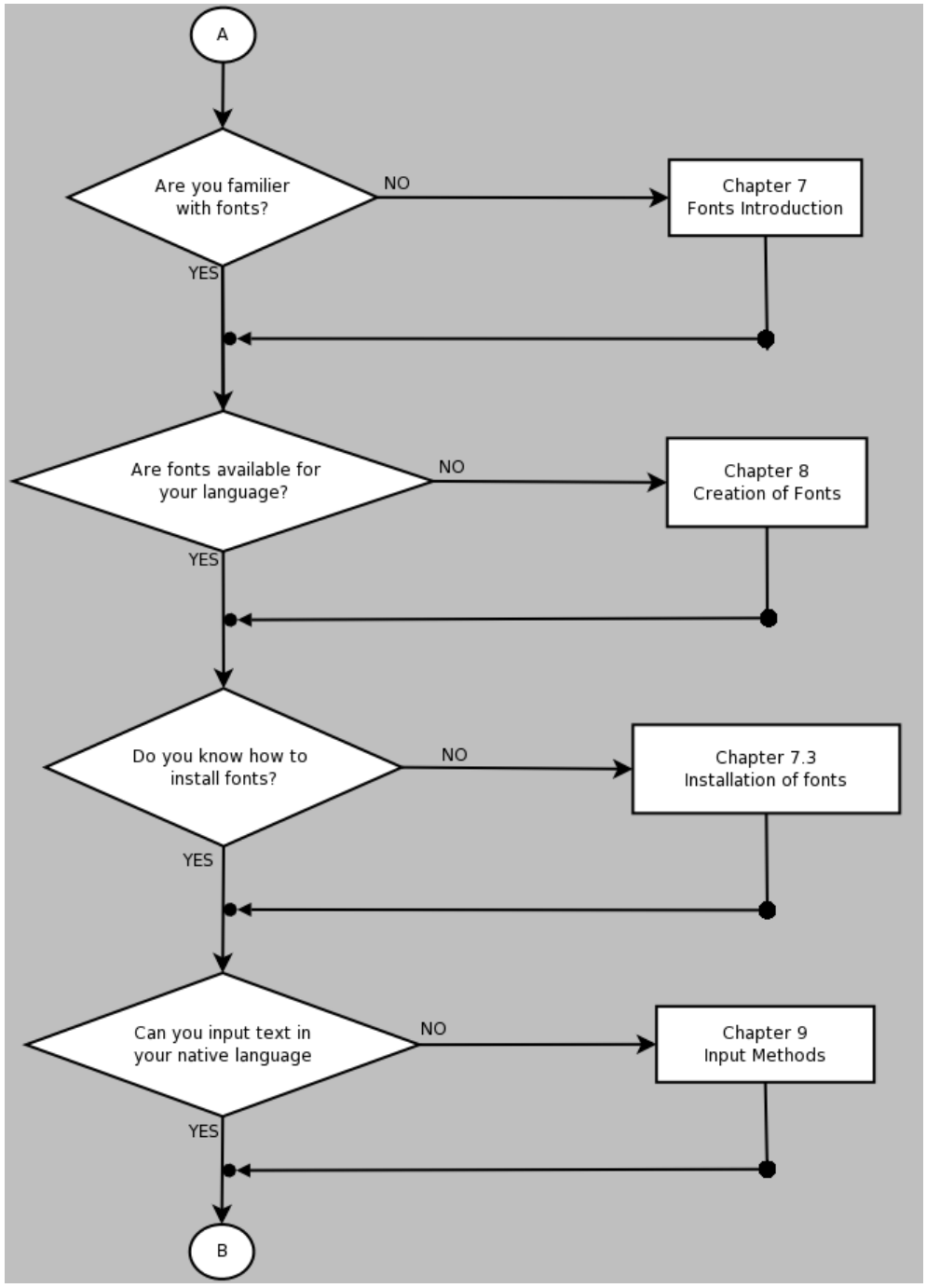
This page is the main reference to the guide. Start from the beginning of the Roadmap section (next), and move down answering the various questions posed. Depending on your answer to these questions, we refer you to appropriate resources (appropriately linked). These resources are also largely organised in a similar need-driven format. When you have completed the branch you took, please use the "back" button on your browser, to come back to previous level (this page or a similar page at the lower level), re-attempt the question and continue. This way, you can get to information as needed, quickly, and will make gradual visible progress through your task of localisation.

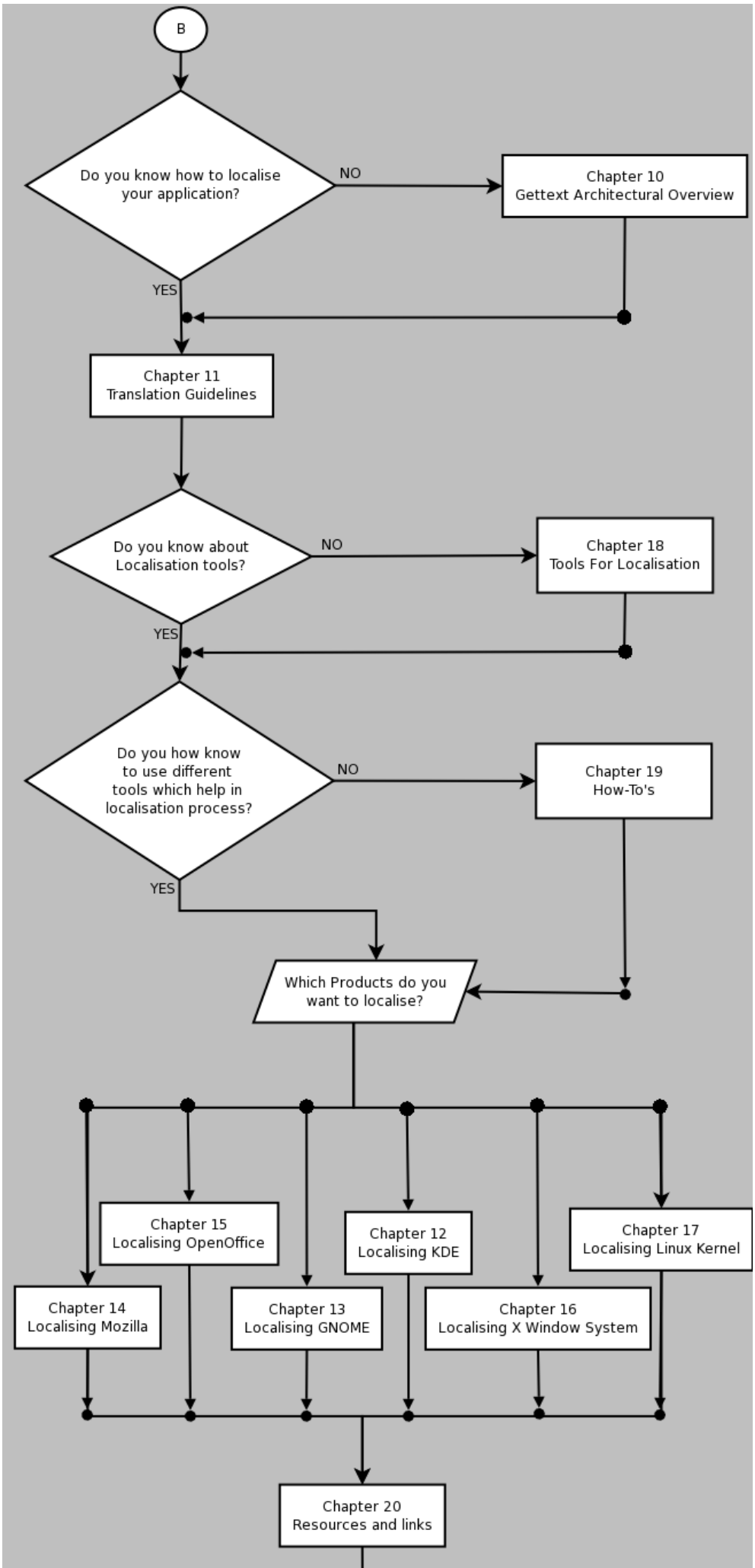
1.3. Roadmap

:Start here: Please go through the following subsections in the order given.

Figure 1-1. Roadmap of Localisation




A
Are you familier with fonts?
NO
Chapter 7 Fonts Introduction
YES
Are fonts available for your language?
NO
Chapter 8 Creation of Fonts
YES
Do you know how to install fonts?
NO
Chapter 7.3 Installation of fonts
YES
Can you input text in your native language
NO
Chapter 9 Input Methods
YES
B


B
Do you know how to localise your application?
NO
Chapter 10 Gettext Architectural Overview
YES
Chapter 11 Translation Guidelines
Do you know about Localisation tools?
NO
Chapter 18 Tools For Localisation
YES
Do you how know to use different tools which help in localisation process?
NO
Chapter 19 How-To's
YES
Which Products do you want to localise?
Chapter 15 Localising OpenOffice
Chapter 12 Localising KDE
Chapter 17 Localising Linux Kernel
Chapter 14 Localising Mozilla
Chapter 13 Localising GNOME
Chapter 16 Localising X Window System
Chapter 20 Resources and links

1.3.1. Setting up the Base

Are you familiar with localisation?

Some general understanding of localisation is a desirable background when undertaking localisation work.

Chapter 3 gives an overview of localisation, Section 3.1 describes the types of localisation

Does it matter which version and/or distribution of Linux I use? What kind of software can be localised?

At present, there are no significant concerns with respect to the Linux distribution or version you use for localisation. Ease of localising the various software depends on a number of factors including its degree of internationalisation, its architecture, availability of prior experience in localising the software, etc. For now, this document will focus on the following software only: Mozilla(Firefox), OpenOffice, KDE/Gnome, X-Window, and Linux kernel. The guidelines may be useful for other software as well.

First stage in localising Linux and associated software into hypolan is to create a locale for hypolan.

Not clear what a locale is? See Section 6.1.

Do you have a locale for hypolan?

See Section 6.3.2 to know how to build a locale.

You now have a running locale setup for hypolan

Next, we need to ensure your computer can display hypolan. We assume hypolan has a fairly accepted script for writing. If it is an oral language (with no script), you need to Chapter 4 for use with computers. Displaying hypolan requires a font and a mechanism to render it. If your computer system can show documents in hypolan, then this step is already done.

Not familiar with fonts? Chapter 7, an overview of fonts will be useful to understand their structure and various types of fonts such as open type fonts (OTF) and true type fonts (TTF).

Is any TTF/OTF fonts available for hypolan? Depending on which version of Linux and X-window system you are using, you need to follow different steps to set these up on your machine.

- TTF fonts and Xfree86 version 4 or higher ? Section 7.3.4.1 describes the installation procedure

- TTF fonts and Xfree86 version less than 4 ? Section 7.3.2 describes the installation procedure
- OTF fonts ? Section 7.3.4.1.1.2 describes the installation procedure.
- Others ? Section 7.3.4.2 describes the installation procedure of all the types of fonts for Xft font system

If there are no fonts available for hypolan, you need to create fonts. Chapter 8 describes the procedure involved and the tools you can use for creating fonts.

In all likelihood, your fonts require only the capabilities provided by the existing framework such as open type or true type, and hence existing rendering engines built into your OS will take care of the display. If this is not the case, you have a more challenging task of developing the rendering engine as well.

As important as displaying hypolan is entering text in hypolan. This requires a keymap, an encoding of the keyboard to the characters of the language. Chapter 9 describes Input Methods

1.3.2. Localising the softwares

Your system now allows text input and output using hypolan. Now you are ready to get into the actual localisation process, per se. The major process in localisation is translation - of the various text pieces you see on the screen into hypolan. This includes the menu items, mouse-over text, labels, error messages, and the documentation. Most of these involve terms with specialised meanings and short phrases, and hence the translations is a difficult task in general. The figures below show the 'gedit' software in English and in Hindi. Figure 1-4 shows 'WELCOME' written in both English and Hindi using 'gedit'

Figure 1-2. gedit in English

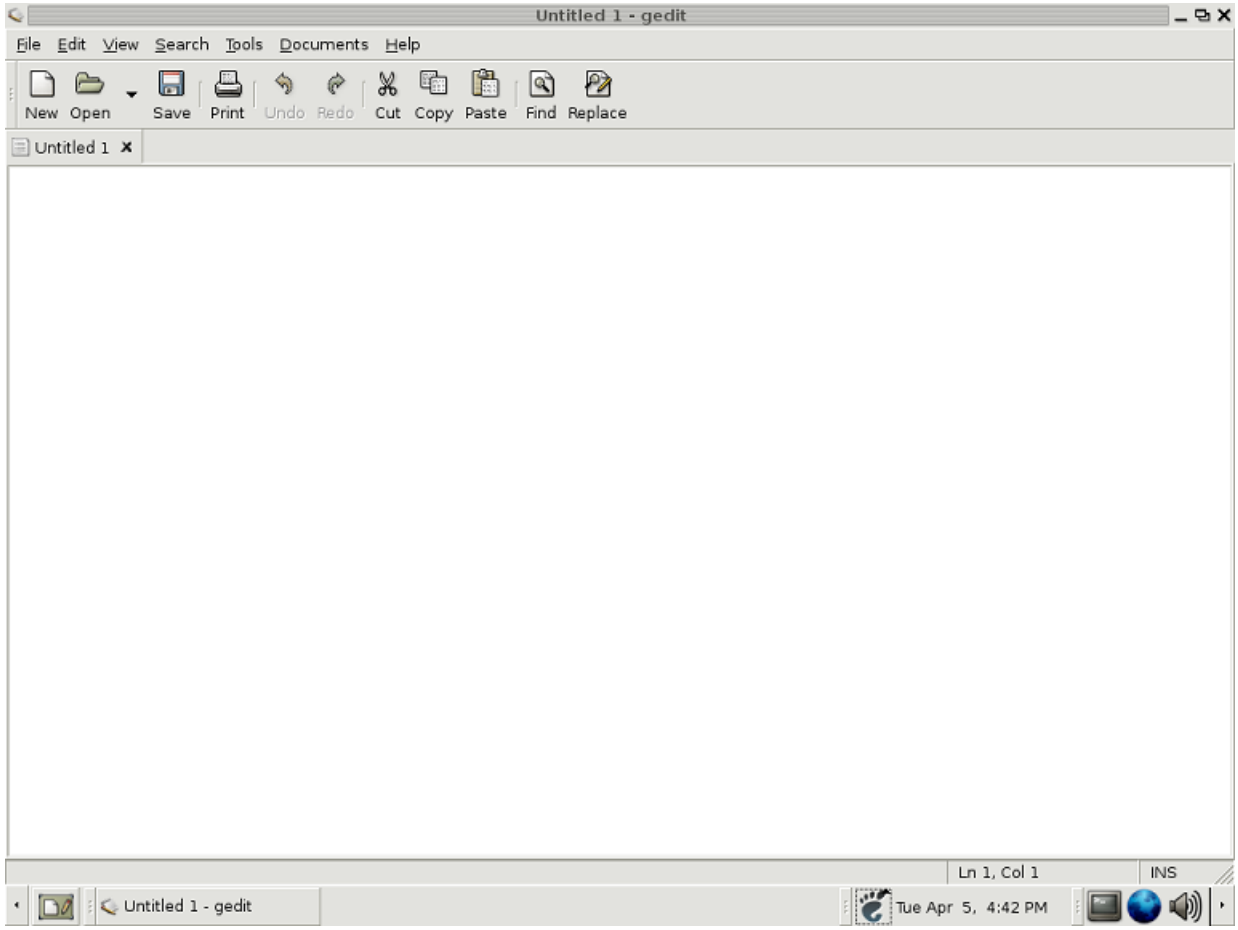


Figure 1-3. gedit in Hindi

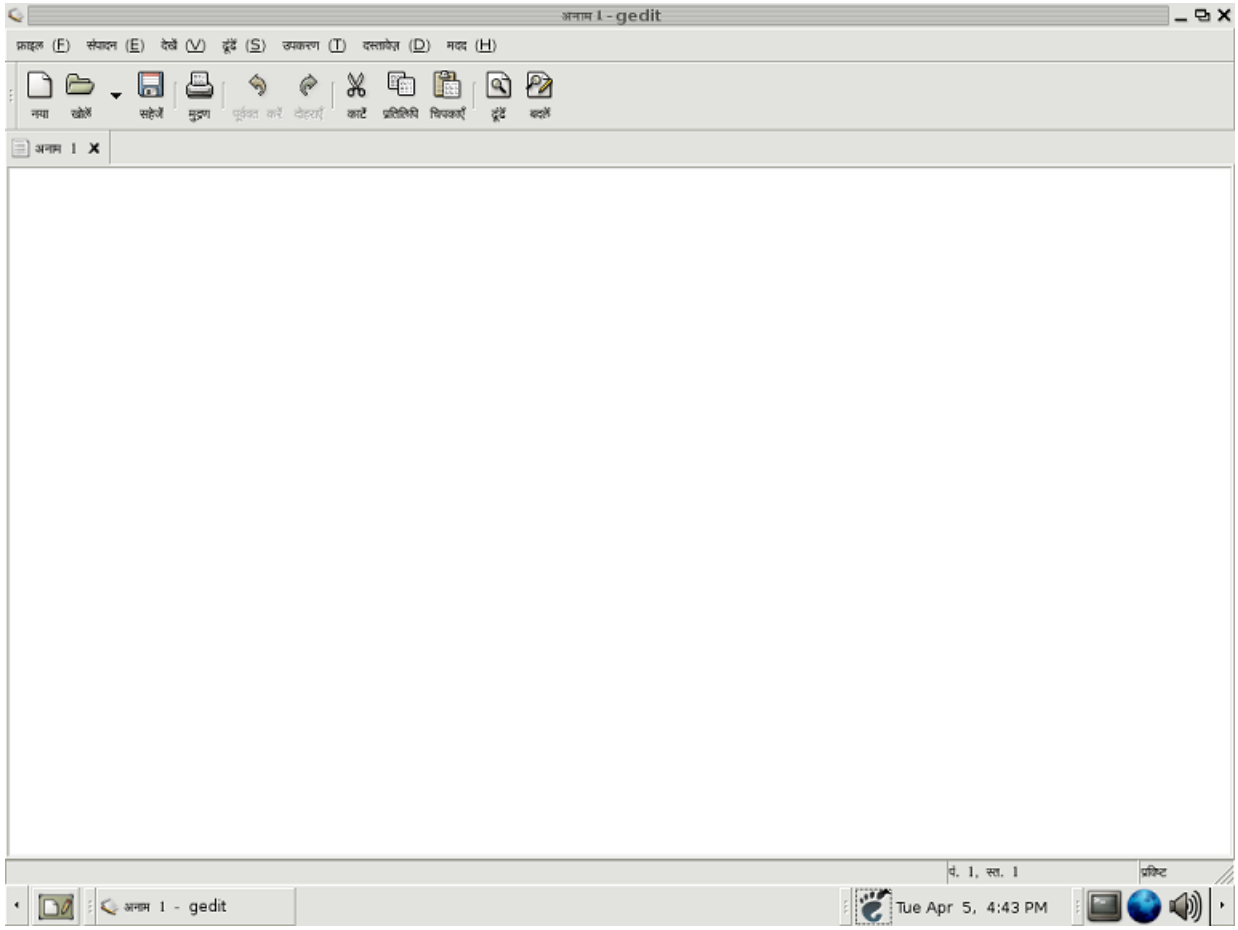
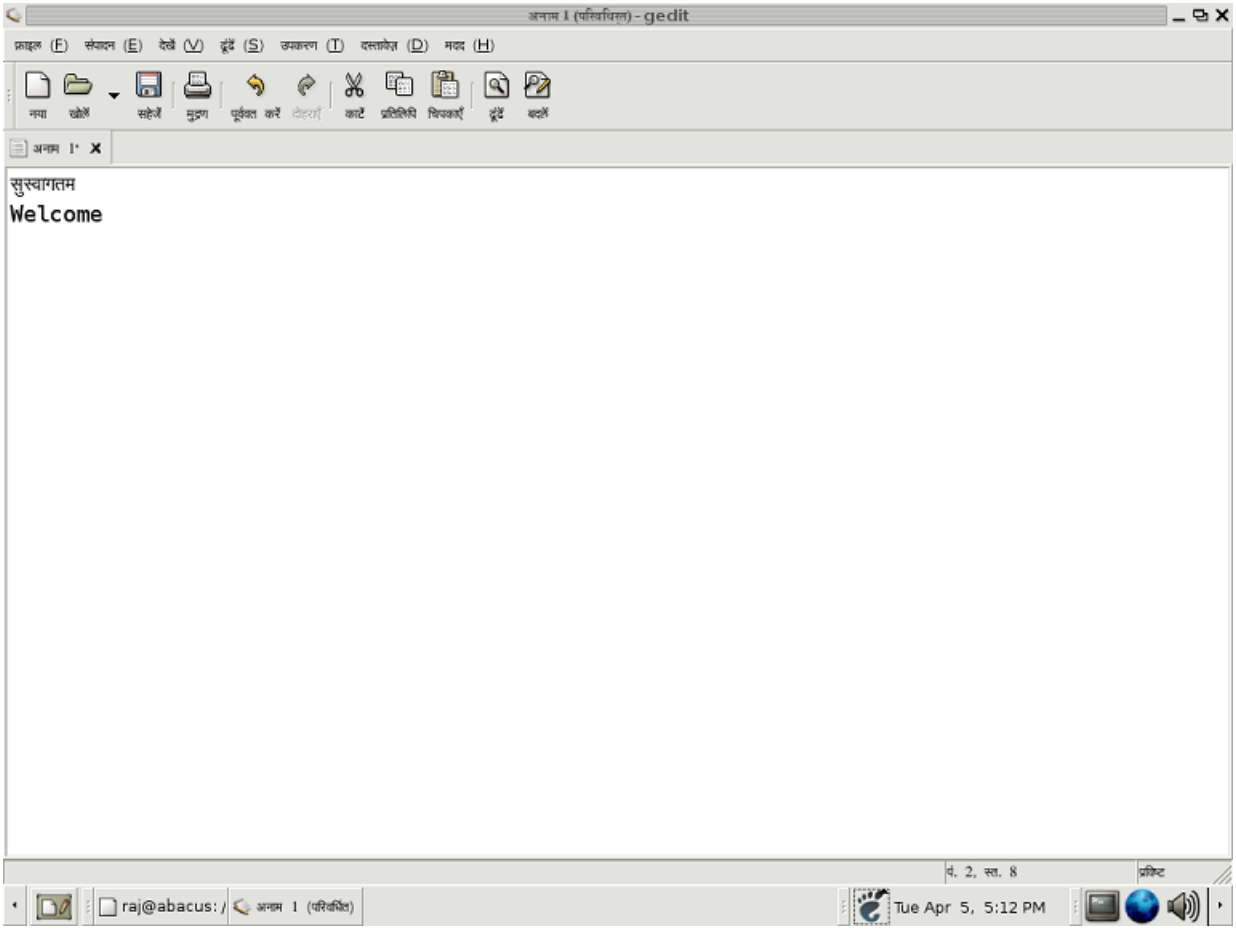


Figure 1-4. gedit used to type 'WELCOME' and its translated Hindi text



Most of the common software you would want to localise are fairly well internationalised. Internationalisation is employed in order to ease the localisation process. More information of internationalisation can be obtained in Chapter 10. This chapter explains how internationalisation is achieved through the gettext framework. Translation in the gettext framework is done using .po files. All you need to do is to pick up these files and insert the translation in hypolan for the strings given there.

It is advisable to go through Chapter 11 irrespective of the software you are trying to localise.

One major concern while carrying out translation is to ensure that the same word/concept is translated to the same hypolan word/phrase everywhere - not only within one application, but across all applications being localised into hypolan. Since, often, a given word in English may have multiple meanings in Hypolan, it is possible for translators to violate this rule, particularly when multiple translators are involved. This vulnerability becomes more significant when the volunteer community is spread across distances, preventing close interaction on a regular basis. Preparation of a glossary containing technical terms in the source language and their equivalent in hypolan is

helpful to avoid these problems. More information on creating and using a glossary could be found at Section 11.2

A variety of tools are available today to help you with the translation process. Some of these provide good GUI for translation of message files in standard formats (eg. .po files). Some provide translation support using a glossary. And so on. Chapter 18 gives the list of tools that help translation

The translation effort for most of the software we consider are high. In most cases, the volume of material to be translated is large, running into tens of thousands of phrases/lines. This necessitates a larger scale effort, preferably involving a community of volunteers. As happens in the open source model of development, a larger community also helps to crosscheck and detect inconsistencies and errors. There are a number of communities already existing for specific types of tasks. The communication model commonly used by these communities is mailing lists, collaboration through editable portals, discussion boards, etc.

The rest of this document is divided based on the software you want to localise.

1.3.3. Localising Mozilla Firefox

Chapter 14 describes localising Mozilla and Firefox, the popular Internet browsers.

1.3.4. Localising OpenOffice

Chapter 15 is a how-to on localising OpenOffice

1.3.5. Localising KDE

Chapter 12 is a how-to on localising KDE

1.3.6. Localising GNOME

Chapter 13 is a how-to on localising GNOME

1.3.7. Localising X-Window

Chapter 16 is a how-to on localising X-Window

1.3.8. Localising Linux Kernel

Chapter 17 is a how-to on localising Linux Kernel

1.3.9. Cultural Localisation of Software

Much of localisation, as practised today, is restricted to displaying all messages and interactions in the local language, hypolan, in our case. Issues such as changing the icons, interfaces, etc to suit cultural variations is the next challenge.

As of now, there is relatively little work being done in the area of cultural localisation. Part of the reason is that these normally involve changes in the program source - sometimes to a substantial extent. In this guide, we will not discuss cultural localisation. For those who are interested in exploring this area, please go through Section 3.1.2.

1.3.10. Validation and Followup

How do I test my system? How do I ensure international acceptance? Any other recommended formalities to be followed?

While it is useful to have a test suite for fonts, localisation process in general etc. , as of now we are not aware of any of these and hence will not be covering test suites in this guide.

1.3.11. Tools for Localisation

Chapter 18 is a list of Tools relevant for localisation, including tools for translation, Font Editors, Text Editors and others.

1.4. Appendices

The appendices in this section contain a compilation of all the relevant material referred to in this document for ease of reference.

Notes

1. The name is derived from 'Hypothetical Language'; but that has no significance in the guide. You can mentally substitute this word by Swahili, Hindi, etc. - whichever language you are interested in.

Chapter 2. Open Source Software

2.1. Open Source Software

This document is not aimed at being a tutorial on Open Source Software(OSS). Please refer to [FOSS1] and [FOSS2] for more detailed treatment of definition, available OSS, development, philosophical issues, etc. For completeness, we briefly describe some of these aspects here.

In the literal sense, the user of an Open Source Software has access to its source code. This gives the user, privilege to participate in the development life-cycle of the application by modifying it, fixing its bugs and re-distributing it. Today two different philosophies exist for this kind of software, namely Free Software (http://www.fsf.org/philosophy/free-sw.html) [FOSS3] and Open Source Software (http://www.opensource.org/docs/definition_plain.php) [FOSS4]. They mainly differ on the amount of freedom the user/developer gets on the software.

2.2. The Development Methodology of OSS

In the earlier days, in the universities and research labs, if an individual gets an idea he/she used to share it with the people around. And the interested people used to work together in implementing the idea. So the work is driven by common interests and is need driven. The roots of Open Source software development came from this.

Later people started sharing the ideas with the entire world. Anyone who was interested could participate in the development. Typically the organisation where the individual was working, would host the source code. So anyone could download the code and improve/customise it. Normally contributions are submitted as patches. The maintainers include that in the source tree after rigorous testing.

2.2.1. The process

For any Open Source project, there will be a few core developers (aka maintainers) and big/small developer base. The Project has its own space in the Internet, where the source code, documentation etc are kept. sourceforge.net, eduforge.net,freshmeat.net etc are popular hosting sites for a large number of Open Source Software. A list of tasks to be done are also kept on the site. Any one can take a task (preferably on prioritised basis), complete it and give it back to the core developers. The core developers review it, check its integrity and then make it part of the source tree. Every project has its own standards including coding standards. The people who want to participate in the development life cycle has to follow these standards.

Now as the developer itself is often a user, he has better understanding of the requirements. The naive users can also participate in the development process by

submitting bugs, by fixing them etc. The fixed bugs are submitted as patches. Once they are reviewed, they will become part of the source tree. So the users using the earlier version of software can apply those patches if they want or they can wait for the next release of the software.

The Concurrent Versioning System (CVS) plays an important role in the development of Open Source software. The versioning of the software being developed is very important as there are a large number of developers involved. For more details on this, go through Guide lines on working with CVS

The resources required in developing Open Source software...

- A central server hosting the source code, preferably running a versioning system,
- A website hosted on the central server,
- Mailing list / discussion board for discussion among the developers,
- Bug tracking system,
- Development tools and environment (including Internet access)

The first four in the above list are provided for free (of cost) by sites like sourceforge, savana, sarovar etc. As the developers are working from all over the world, they have their own machines and resources. Normally the tools used for development are Open Source software itself and are free (of cost too). Internet access is a must for the user to be part of the Open Source community. We can't even think of Open Source software development without Internet.

2.3. Why OSS?

The interest in OSS in most communities, particularly in the developing countries, comes from a few major perspectives. These include the contribution to making computing affordable (in terms of cost), and accessible (in terms of usability). These are described in more detail below. In addition, there are also the advantages of providing more options for the user, the ability to leverage local and global expertise for solution development, and being able to contribute back to the international community our own developments.

Free access to source code also enables collaborative development (including testing and bug fixing) - a methodology that has become characteristic of open source developer community. Individuals and their ideas are more visibly recognised in the open source model. Almost all major open source systems have an associated 'inventor', unlike the proprietory systems, which are known essentially by the name of the vendor. Many feel that this adds to the strength of the OSS model by recognising the individuals for the contributed intellectual property.

Affordable Computing

For the common man as well as the academic and public sector institutions, the interest in the OSS model, is primarily because of the perception of a low cost of ownership. Two major factors translate to substantial reduction in cost of building software solutions, namely, that much of the OSS is available free of cost, and that one could modify or adapt them to one^Ös requirement without violating any conditions from the developer/vendor. When purchased, the basic software for any computing system, such as operating system and office suite cost many times the annual income of an average citizen in the developing countries. Once these costs are reduced to almost nothing through the use of open source software, the computing cost reduces to primarily the hardware cost. Open source community has been able to show convincingly that despite being free, the software achieves a reliability and usability rating as high as commercial softwares of a similar nature. Even mission critical applications are today running on open source solutions.

Accessible Computing

As computing technology plays a utility role as an essential component of a good quality of life, it is important that cultural and language barriers do not prevent proper use of such technology. This requires softwares of wide interest including e-mail management, word processors, browsers, spread sheets, entertainment tools, etc to provide a comfortable interface that suits its users. All software developers, today, recognize the need for tailoring (localise) software systems to suit the linguistic and cultural requirements of various communities. Commercial developers and vendors can afford to do this only for large communities. Localization efforts have borne more fruits in the case of open source software systems, compared to proprietary solutions. For example, while Windows offer support for about 20 languages, the popular OSS desktop environment KDE is today available in over 40 languages with another 40 or so in the pipeline. Similarly the OSS Officesuite supports over 30 languages including some Indian languages, with over 40 languages in the process of being supported. These include some of the rare languages spoken or used by small communities - unlikely targets for proprietary developers. The access to source code and a wide community of developers knowledgeable about the source, are major factors in enabling large scale efforts to localise the system for various communities. Even relatively small communities are able to enter into such efforts, which would not be viable for commercial vendors to undertake.

Bridging the Digital Divide

A combination of accessible and affordable computing, when seen in the light of a large country like India, shows the relevance of OSS in bridging the digital divide. Even at the best price in the market, computing systems will be beyond the reach of majority of Indians, given our per capita annual income of less than Rs 20,000. Open source and freely distributable software reduces cost of computing systems significantly. Coupled with movements towards low-cost PC, we can now aim to have computer systems which are affordable to much larger Indian segments.

India speaks scores of different languages, all quite different from English. The English speaking community in India is not even 10% of the population. Among the various languages, some have relatively small population base. Reaching out to communities who are using relatively less common languages is possible only by localization efforts put in by volunteer groups. This, in turn, is realistic only in case of open source software. Along with languages, the cultural diversity in India is also very high across its vast geographical breadth and length. Conventions (including symbols, gestures, etc) in one part of the country may convey a different interpretation elsewhere. These impact the use of messages, icons, and interaction in software systems. While localisation efforts are yet to address such cultural variations on a large scale, this is a strategically important emerging area, as information and communication technologies are reaching out to the masses.

ICT as a Base Necessity

Information and communication technologies are gradually becoming an essential ingredient in one's life, irrespective of one's education, career and life style. As an important enabler for quality of life, it need to be available widely without restriction. One could, perhaps, compare this to utility infrastructure such as roads. It is important that these are available easily and at affordable terms. Proprietary vendors, with business and profit as the bottom line, can not be relied on to provide this. OSS model is, so far, the only viable model. It also encourages volunteer efforts to develop the resources further to meet short falls as and when they are noticed.

Reliability of Systems

OSS model contributes to the general security of systems in two significant ways. Firstly the deploying organization has access to the source code, and hence can, if needed, study the same to find out possible risks and unauthorized modifications. There is also the option of modifying the source code, if adequate competency is available, to correct any potential risks. Secondly users are not dependent on a system owned and controlled by one particular vendor. A wide community of knowledgeable people ensures more reliability and sustainability for the system than would be possible in the case of proprietary software.

At a national level, this can be viewed from a broader angle. Having much of its critical computer systems work on proprietary platforms, makes the entire infrastructure vulnerable to the corresponding vendors. As new versions of software are released, old versions are no longer supported, and users often have no option but to migrate to the new version irrespective of his/her interest in the new version, for fear of losing support. While they may provide localised versions of their products today, there is no guarantee of all their future offerings continuing to do the same. With open source software, one has the option of undertaking this work, on one's own, if required.

Less Dependence on Imports

Much of the software platforms and systems in use across the world are developed in

the developed nations, predominantly in the US. All other countries using these softwares, have to purchase licenses from the developers. With more and more nations going for e-governance and related automation of administrative functions, the expenses involved in acquiring software licenses can be high. This can be a substantial drain on the exchequer for most developing and under-developed nations. This money would, otherwise have been spent on development work for the people in that country.

Opting for open source software substantially reduces the cost of purchase. Of course, one has to consider the cost involved in paying for support and maintenance. This may not be a small amount, particularly for open source software, since there is no 'vendor' support available. However, support and service for open source software only requires skills and experience, and will often be available within the country. Thus, this cost, wherever incurred, is an internal cost and indirectly helps to develop and sustain local industry for providing such support/service. This knowledge can be of value to other organisations as well. Thus, unlike, outright purchase of software from abroad, these expenses have an overall positive contribution to the economy of the country.

There is a related positive effect stemming from this line of thought. Access to freely distributable and modifiable source code enables local development groups to enter into product/system development more easily. For example, one could develop specialised or enhanced versions of an existing system, without having to build the entire system from scratch. Therefore, local software development efforts and hence the local software industry, can enter the IT market through a much lower entry barrier. This, in the longer run, provides for growth of local industry, retention of skilled human resources, more training opportunities, etc.

Interoperability and Vendor lock-in

In areas such as information technology which are charaterised by high rate at which systems and platforms become obsolete, it is important for information in the form of data, documents, etc to be stored in open publicly known formats. This is the key to interoperability - the freedom and flexibility to change between different software packages, platforms and vendors. The interoperability frees users from vendor lock-in where one is forced to go along with a particular system because a lot of resources are already in their proprietary format and conversion is too costly in time and money.

According to a study carried out by the Swedish Agency for Public Management, "it is also necessary to place demands on open standards and file formats in order to achieve interoperability between different systems." Consider, for example, the huge amount of e-learning content being prepared through various efforts at many places around the world. Much of this content is prepared in proprietary formats such as MSWord, Powerpoint, Toolbook, etc. These formats are largely determined and sustained by the associated vendor products. Imagine the situation when one or more of these are no longer supported by the corresponding vendor. The cost and time requirement for converting these into some other format can be unaffordably high.

Most documents prepared in electronic form today are likely to be in MSWord format,

and presentations in Powerpoint format. Many web pages today use browser specific features, preventing users from using other web browsers such as Mozilla.

Open source community has generally opted for open file formats and standards, and hence open source software provides a much higher level of interoperability compared to proprietary solutions. The cource code availability enables people to document standards used by a system (by reverse engineering, if needed) thereby making the standards open. Compliance to known standards is, perhaps, a natural companion of the OSS work culture anchored on sharing and working together.

Chapter 3. Localisation Process

3.1. Background of Localisation

Read if: you are not familiar with what localisation means and/or want to know about localisation in general. Recommended if you are undertaking any localisation work.

Localisation (l10n) is the process of adapting, translating and customising a product for a specific market. The Localisation Industry Standards Association (LISA), defines l10n as "Localisation involves taking a product and making it linguistically and culturally appropriate to the target locale where it will be used and sold."

Localisation is broadly classified under two categories:

- Text Localisation
- Cultural localisation

3.1.1. Text Localisation

Text localisation deals with presenting menus, instructions, error messages and all other text messages in relevant languages. One area of concern in text localisation is the possible expansion of text when the text is translated to the local language. The average length and width of any statement is greater in any non-English language as compared to the English language. For example the statement 'unable to open file' in Hindi will be

फाइल खोलने में असमर्थ

Thus, as seen in the example, the expansion can be both horizontal and vertical.

Text localisation should be applied to documentation also. Thus one may need to more translation of more than thousand pages of text which could be very painful. For such translations, internationalisation comes in handy. [Internationalisation process is beyond the scope of this guide and hence will not be covered in detail.]

Text localisation becomes difficult when the text directionality of the source and target languages are different. For example, Hebrew and Arabic texts are read right-to-left while English texts read left-to-right. Also sometimes within texts that read right to left there can be some part that can read from left to right. An example of this is an Arabia document showing an English clipping. This could cause more problems.

Many Asian language scripts have characters that represent some idea or phrase rather than a single letter. This increases the number of characters making the input process very tedious.

Text localisation would involve translation or transliteration of words. It is not always a good practice to translate all the words. It maybe best to transliterate to some words, especially technical words, for example the word "mouse". To know more about which words should be translated and which should not, refer to Chapter 11.

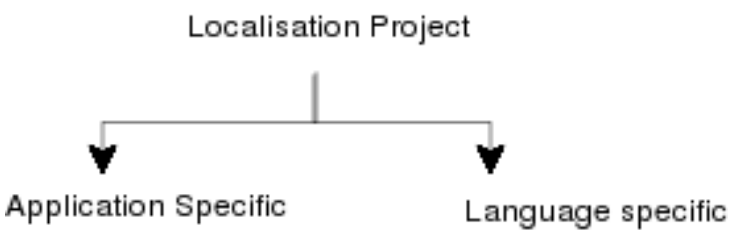
3.1.2. Cultural Localisation

Cultural localisation is concerned with adapting the software to the cultural conventions of the target user community. It worries about which colours to use, which icons to use, the message conventions used etc. This is a very important aspect to localisation as different colours and icons signify different things in different regions and proper message conventions is a must for mass acceptability. For example, the red colour signifies Danger in US while it denotes Happiness and Good luck in Asian countries. Similarly, white signifies Purity and Cleanliness in the U.S. while it signifies Death in Japan. Same is the case with icons. An example to show this is the famous Apple macintosh trash Can icon which is used when a user wants to temporarily place files and folders for future deletions. In Britain, however, this icon resembled more to a postal box than a trash can. Thus the icons which are chosen for any software application must also be very well thought-out. If you want to know more about Cultural localisation you can read International User Interfaces (Edited by Elisa M. del Galdo and Jakob Nielsen) or the paper 'Internationalisation and Localisation of Software' (by George Calzat).

3.2. Planning Localisation Project

A Localisation Project can be classified into two categories as shown in the figure below:

Figure 3-1. Types of Localisation



Application specific localisation means, taking one particular application, like Mozilla, and localising it into different languages. If this is the aim of your project then you will have to first decide the application you want to localise and the languages you want to localise the application in. Having decided this, you will have to identify the translators for the project. You can choose to have translations in different languages running in

parallel or decide on taking one language at a time. The number of translators you hire might depend on this and this may affect your Project budget.

Language specific localisation means localising some set of applications into your local language. In this case you need to first decide on the set of applications you want to localise. You, may prioritise applications in a certain order to carry on your localisation task. For example, you could decide that you would first localise the OpenOffice suite, then a web browser like Mozilla, next a desktop environment and so on. This priority order can be decided by taking into account the applications most used in your region. Also if you are localising OpenOffice you may choose to not localise AbiWord.

Once the applications are decided and the priority list is prepared you could choose to work on localising a certain number of applications concurrently depending on the number of localisers (mostly translators) on the job. Issues such as these can be decided by taking the project budget into account.

Having decided upon the type of your localisation project, the next step will be to identify the people involved in the project and the overall activities with their timelines. The following table gives a list of common categories of people involved in a localisation project[MSL1]. As it generally happens, one person may play more than one of these roles. The table illustrates the various activities and roles relevant in a localisation project. This table also helps you to plan your human resources, if you are looking at deployment of large number of human resources (volunteers/professionals) or if you are considering a large localisation effort.

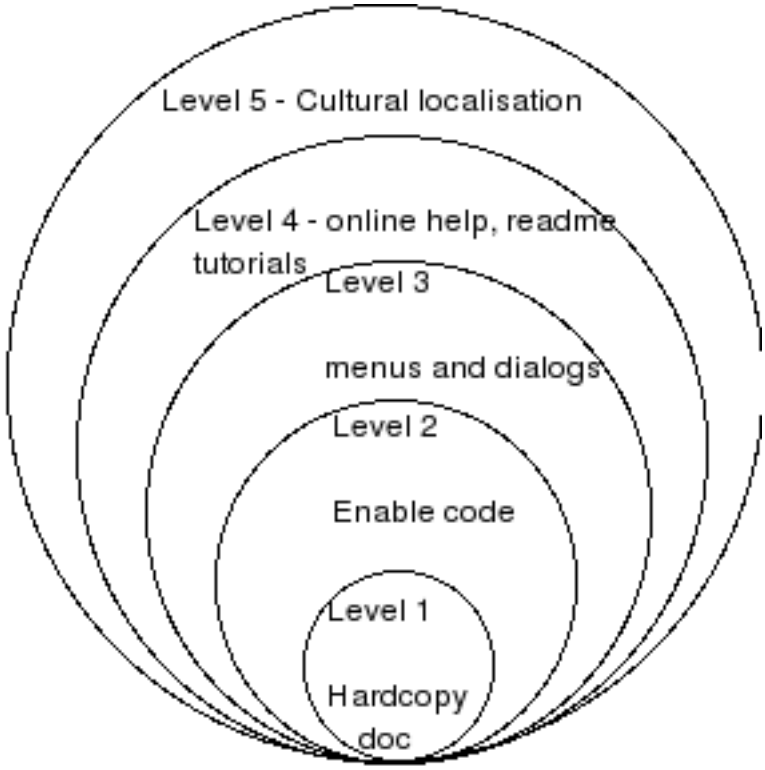
Table 3-1. Categories of people involved in any Localisation Project

Persons involved	Abbreviation
Localisation engineer	LE
Localisation Manager	LM
Translator	TR
Testing engineer	TE
QA Specialist	QS
Proof-reader	PR
Linguist	LG
Web Designer	WD
Web Manager	WM
Web-site Builder	WB
Client	CL
Font Designer	FD
Desktop Publisher	DP
Localisation Vendor	LV

Scope of a localisation project can be divided into four levels[MSL1] as shown in the

figure below:

Figure 3-2. Localisation Levels



- Level 1 - Translate hard-copy documentation : At this level, the localisation process involves translating the hard-copy documentation which may include user manuals, programmer's guide, installation guide, marketing pieces etc. Since the application is not localised the screenshots and other images would remain in the original language but the written text would be translated. The localisation activities at this level can be summarised as below:

Table 3-2. Activities in Level 1

Activity	Persons involved
Collect the materials to be translated along with specifications	TR
Analyse the content and estimate the cost based on the amount of translation, number of translators involved and cost of printing material	TR,LM

Activity	Persons involved
Assign translators and explain the technical issues involved and the context of translation.	
Collect a dictionary for translation, prepare a glossary of commonly used technical words, leverage translations	TR
Print translations	DP
Proofread translations	TR and experts
Test the translations for completeness and correctness	TR,CL

- Level 2 - Enable the code: At this level, localisation involves altering the source to handle input of local language text, display local language text and editing and printing local language text. This level involves no translation cost but has development cost. The following table lists the activities involved at this level:

Table 3-3. Activities in Level 2

Activity	Persons involved
Take care of character encoding related issues (Don't know what is Character Encoding. Refer Chapter 5.)	LE
Develop local language fonts and display layouts. (Don't know how to create fonts. Refer Chapter 8.)	FD
Develop rendering engine to display the fonts.	LE
Create your own locale. (Refer to Chapter 6 to know all about locales or Section 6.3.2 to know about how to create locales)	LE
Modify source code to enable local language input, display.	LE
Integrate rendering engine with the software and test	

- Level 3 - Translate software menus and dialogs: At this level, you opt for translating only the user interface.

  The major tasks at this level include huge amount of translation of message strings and menus which are a part of the GUI. The LM has to be very sure that the translator gets all the specifications and the context correctly.

- Level 4 - translate online help, tutorials and README files: This can be considered

as an extension to Level 3. Having translated the software menus and dialogs, the next step would be to translate the online help files. The major cost involved at this level is translation cost.

- Level 5 - Translate culture specific features: At this level you take care of Cultural localisation issues. This level requires you to analyse the cultural conventions of your target community.

Thus, while planning your localisation project you need to decide the scope (in terms of level 1..5) and apply timelines accordingly.

3.2.1. Community model and CVS model

You must have observed in the list of localisation activities described in the previous section that translation is one of the major activities in almost all levels of localisation. Different people follow different approaches for translation. In this section we are going to discuss two approaches to translation : the approach followed by the localisation community and the so-called non-community model.

3.2.1.1. Community Model

The localisation community follows what is called the "CVS" approach. In this approach, the files for translation, for example the PO files (See Chapter 10 for information about PO files), are taken from a common place, translated and then kept back at that place. By doing so the translations are available to everyone on a regular basis and in the future versions of the system. To explain with an example, consider that you are planning to localise the OpenOffice into hypolan. You get the string files for translation from their CVS (The procedure to obtain the files are explained in detail in Chapter 15). These string files contain the messages in the original language and a place to specify the translation of the message. You need to write the translated hypolan text for all the messages in the string file and save it back to the CVS (The exact procedure to do all of these is explained in detail in Chapter 15). Once this is done the translations would be available for anyone else who are trying to edit the file and will also be available in all future versions of OpenOffice.

The community model provides for effectively leveraging the work done by others and also contributing your work to speed up the process for others still coming along. It is also common that multiple groups are working on the same language with limited human resources. In such cases the periodic sharing of one's work helps to allocate available resources more effectively and reduces duplication. It also enables you to get quick and early feedback on your work. Thus the community model is quite popular among the open source community initiated localisation.

3.2.1.2. Non-Community Model

Some people follow the approach of downloading all the string files, making the necessary modifications to it and make a localised application of them. But they do not

upload the translated files back to the CVS. An example of such an approach is the BharateeyaOO [OOL10]which is the Office suite localised in Indian languages which can be used across all major platforms. This is a separate project of OpenOffice in the local languages. To know more about BharateeyaOO you can refer BharateeyaOO (http://www.ncb.ernet.in/bharateeyaoo/) [OOL10]. In this approach, if OpenOffice comes up with a new version then the changes made in BharateeyaOO will not be reflected. Also, if some other translator wants to translate OpenOffice in an Indian language then there is a possibility that he may not be aware of the BharateeyaOO project or will not know whom to approach for the string files and hence may start with the translations all over again.

This model helps you to have higher control of vocabulary and any other program changes you make to the original system.

Thus, as seen in the preceding sections, planning your localisation project needs you to consider a lot of aspects and is not a trivial job.

Chapter 4. Evolving a Visual Script

For using computer in your native language, your native language need a script for obvious reasons. If your language does not have a script, you will have to adapt/create one. Since this is a relatively rare situation in localisation, we are not discussing this issue in detail here.

If you are creating a new script, you will have to create fonts for that, provide rendering engine support, etc.

Alternately you can adopt some other language's script. That is you will be using their alphabet to write your language. You will have to use that particular language script as your script for the translations. Note that, you don't have to create separate fonts, give rendering engine support etc. Only thing you will have to do in this case is transliteration.

The meaning of transliteration from dict.org.

```
Transliterate: To express or represent in the characters of another
               alphabet; as, to transliterate Sanskrit words by means of
               English letters.
```

Please note that you can't use the translations of that language even though you are using its fonts, rendering module etc.

Chapter 5. Character Encoding

5.1. Introduction

Read if: you are unfamiliar with notions of character encoding and terms like UTF-8. Recommended if you are undertaking any localisation project.

Computers are very powerful devices with a significant limitation: they deal only with numbers. So whenever we want to store characters used in human languages, in a computer, we need to store it in a way in which the computer understands. This means that we need some kind of mapping in which a particular character corresponds to a particular number. This mapping, loosely, is known as character encoding.

Developing an application which would be used internationally requires the application developer to cater to character sets of various different languages. This requires the application to be able to represent characters of different languages. When localising a software into a specific language we need to ensure that the software uses an encoding scheme which can represent characters of the target language. While a properly internationalised software may provide this support, in other cases the original software may need to be extended suitably.

Every language is characterised by a set of characters. A character is the basic unit of text. Characters are used to form larger textual units like words.

The first step in representing human language in a computer is to identify the characters in the language and collect all of them to form a set of characters called the Character Repertoire. Every language will have commonly used characters, characters that are not so commonly used (for example, an ancient language could have some historic characters which will not be used frequently), characters with distinct meanings like the punctuation characters, characters that denote numbers, vowels, consonants etc. In addition to this, there could be multiple variants of a character in different regions. Which of these characters should go into the Character repertoire and which should not has to be decided. For example, if there is a character with multiple variants then it has to be decided whether to include all the variants or just one form of the character in the Character Repertoire.

Once the Character Repertoire is formed, the next step is to define an encoding scheme which maps each character in the Character Repertoire to a unique integer, the mapping being called the encoding. Encoding schemes refer to this unique integer as the code-point.

One thing to be noted here is that the Character Repertoire determines what languages would be covered by the encoding scheme. Whichever languages need to be covered by

the encoding scheme the characters of all those languages should go into the Character Repertoire of the encoding.

5.2. Properties of character encodings

5.2.1. Fixed Character Repertoire encodings and Open Character Repertoire Encodings

For most of the character encoding schemes, the Character Repertoire is fixed, i.e., once the repertoire is decided upon it cannot be changed. Any new addition to the Repertoire would amount to creation of a new Character Repertoire which will be given a new catalogue number.

On the other hand, there are encoding schemes which work on Open Character Repertoires, wherein addition of new characters would not amount to creation of new Character Repertoires. Different encoding schemes have their own ways of extending their Character Repertoires. Some extend their Repertoires periodically while others version their Repertoires by publication of editions.

5.2.2. Stateful and Stateless encodings

Encodings can be classified as stateful or stateless.

An encoding is said to be stateful, if the meaning of any character depends on the characters already encountered. For example, the Shift-JIS encoding used to encode the Japanese language is grouped into multiple sets, each set containing less than 256 symbols. To switch between appropriate data sets, special escape characters are used in the data stream. Therefore, identification of the relevant data set and thus the actual characters represented requires knowledge of the escape characters earlier encountered.

Stateless encodings are those in which the code-point of a character is not affected by the symbols that precede it in the data stream.

5.2.3. Single-Byte vs Multi-Byte encodings

Encodings are also classified as Single-byte or Multi-byte encodings.

Single-Byte encodings

Single-Byte encoding schemes use a single byte to represent each character. They are the most efficient encodings available. They take least amount of space and are easy to process because one character is represented by one byte. 7-bit encoding schemes and 8-bit encoding schemes come under this category.

A 7-bit encoding scheme can define up to 128 characters and normally support a single language. An 8-bit encoding scheme can define up to 256 characters and generally support a group of related languages.

Multi-Byte encoding

Multi-Byte encoding schemes use multiple bytes to represent a single character. These schemes use either a fixed or variable number of bytes to represent a character.

A fixed-width Multi-Byte encoding scheme uses a fixed number of bytes to represent to represent every character of its Character Repertoire. Examples of fixed-width encodings are UCS-2 (a 16-bit encoding), UCS-4 (32-bit encoding), UTF-32 (32-bit encoding).

A variable width Multi-Byte encoding scheme uses variable number of bytes to represent different characters, .i.e., some characters are represented with 2-byte codes and some take 3-byte codes. Some Multi-Byte encodings use certain bits to indicate the number of bytes used to represent the character. UTF-8 and UTF-16 are examples of variable-width Multi-Byte encoding forms.

5.3. Different Encodings: A Quick Tour

There are various different Encoding systems in use today. These include ASCII which is a 7-bit encoding system, the code-pages defined by the ANSI standard, UCS-2 and UCS-4 defined by the ISO 10646-1 standard, ISO 8859-1, Unicode and many more. The major encoding schemes in use are discussed in detail in this section.

5.3.1. ASCII

ASCII was invented in the early days of Unix and C. The only characters that mattered then were the English characters. These characters were encoded using ASCII. In its initial days, ASCII had competitions from other encoding systems like EBCDIC and Baudot. But eventually ASCII dominated and EBCDIC and Baudot are rarely used today.

ASCII is a 7-bit encoding system, in which every character is represented using a number between 32 and 127. With 26 upper-case letters, 26 lower-case letters, 10 digits and about 20 punctuation marks, this range is used up. For example, space is represented by '32' , the character A is represented by '65' and the character a is represented by '97'. The codes below 32 are called unprintable and are used for control characters. For example, code 7 makes the computer beep and 10 is the newline character.

5.3.2. Base64

This is an encoding scheme in which binary-encoded data is converted into printable ASCII characters. The only characters used in this encoding are upper and lower-case Roman alphabet characters (A-Z, a-z), numerals (0-9), '+' and '/' sign with '=' as a special suffix code. This encoding scheme is used for sending binary data (e.g. Microsoft

Word document, images, etc) using media such as e-mail which normally accepts only printable text.

5.3.3. CODE-PAGES

ASCII was invented at a time when most of the computers used 8-bit words. Hence computers could not only store any given ASCII character in one word but also had 1-bit to spare. Since the extra bit was getting wasted, the computer users thought that it could be used to create more codes between 128 and 255 and thus create encoding schemes that could operate on a bigger Character Repertoire. This led to creation of the Original Equipment Manufacturer(OEM) code-pages.

IBM came up with a character encoding scheme called the OEM character set which provided space for some accented characters and line drawing characters like horizontal bars, vertical bars, etc. in addition to the characters covered in the ASCII Character Repertoire. These were used for creating good user interfaces in those days. But the problem arose when people outside America started using these PCs and came up with their own OEM character sets.

In each of these character sets, the first 128 character codes were same as ASCII character codes and the codes above 128 were specific to individual encoding systems. Thus, on PCs in America, the character code 129 would display an accented é while on computers sold in Greece it was a German character λ. Hence when an American sent data 'daté' to the Greeks, it would arrive as 'datλ'.

Eventually, all the OEM character sets got codified in the ANSI standard, where a uniform character set was accepted for codes below 128. However, for codes from 128 to 257 a set of different mappings were accepted. These systems were called code-pages. Thus Israel DOS used a code page numbered 862, while Greek used 737.

Using code-pages helped when people wanted to use characters of one particular language in addition to English. But problems arose when more than one language was required to be used on the same PC and for languages with large character sets. For example, getting Hebrew and Greek on the same computer was impossible as they required different code pages representing different characters for codes above 128. To make things worse, people in Asia came up with encoding systems in which some letters were stored in one byte and others in two. Due to these problems this encoding is rarely used today.

5.3.4. ISO 8859-1

ISO 8859-1 is a character encoding scheme defined by the International Organisation for Standardisation (ISO). ISO 8859-1 is an 8-bit character encoding and its Character Repertoire contains all the characters useful for the West European languages. The languages supported by ISO 8859-1 are Afrikaans, Basque, Catalan, Danish, Dutch, English, Faeroese, Finnish, French, Galician, German, Icelandic, Irish, Italian, Norwegian, Portuguese, Spanish and Swedish.

ISO 8859-1 is one part of the ISO 8859 standard which specifies many Character Repertoires from 8859-1 to 8859-10. These Character Repertoires include characters of different languages. All these Character Repertoires use the characters 0xa0 through 0xff to represent national characters, while the characters in the 0x20-0x7f range are those that are used in the ASCII Character Repertoire. Thus ASCII Character Repertoire is a proper subset of ISO 8859-X Character Repertoires.

ISO 8859-1's character set is used by AmigaDOS, MS-Windows and practically all Unix implementations. MS-DOS, however, is not compatible with this character set.

Though ISO 8859-1 is an international standard, not all use this encoding. Many computers use different vendor-specific encodings. In order to edit or view files written in different encodings, one has to first translate them to ISO 8859-1 based representation. There are several translators available for free on the Internet for this purpose, the most famous among them being Recode. Recode is available from ftp://prep.ai.mit.edu/u2/emacs

5.3.5. UCS

The international standard ISO 10646 defined the Universal Character Set, also known as UCS. UCS is a superset of all the other character set standards mentioned above. It contains the characters needed for practically all known languages.

ISO 10646 defines a 31-bit character set, allowing a maximum of 2147483648 number of characters. In this set, all the characters used most commonly, including those found in the earlier encodings are defined in the first 65534 positions. This 16-bit subset of UCS is called the Basic Multilingual Plane (BMP) or plane 0. The characters that were later added outside the BMP were generally characters used for specialist applications such as historic scripts or scientific notations.

UCS assigns each character a unique code number (known as code-point) and also an official name. Thus 'A' and 'a' will have different names and code numbers. All UCS code numbers generally start with 'U+' (e.g. U+0041 for the character 'A'), although "U+" is just a representation convention. The UCS characters in the range U+0000 to U+007F are identical to ASCII and the characters in the range U+0000 to U+00FF are similar to ISO-8859-1.

Some code points in UCS are assigned to Combining Characters. These are not actual characters . They are some accents which are used with characters.This way one can place an accent on any character. For example, ´ is not an actual character but can be used with characters a, e, etc. to give different characters.

There are three implementation levels of UCS. This is because not all systems can support all the advanced features of UCS. Hence ISO defined three implementation levels, as follows:

- Level 1: Combining characters and Hangul Jamo characters are not supported in this level. Hangul Jamo characters are required to fully support the Korean script including Middle Korean.
- Level 2: These are similar to level 1 except that a fixed list of combining characters are now allowed (eg. for Hebrew, Arabic, Devanagari, Bengali, Gurumukhi, Gujarati, Oriya, Tamil, Telugu, Kannada, Malayalam, Thai and Lao).
- Level 3: All UCS characters are supported.

5.3.6. Unicode

Unicode is an effort of the Unicode Consortium to create a single Character Repertoire covering all the possible characters across the world. Unicode Consortium is a non-profit organisation founded to develop, extend and promote use of the Unicode standard.

The way Unicode looks at encodings is different compared to ASCII or the code-pages of ANSI.Unicode assigns a unique integer value (like U+0065) for each of the characters in the Character Repertoire. This unique integer is called the code point. These code points are then mapped to a sequence of code units, where a code unit is a unit of memory: 8 bits, 16 bits or 32 bits. This mapping is called the Character Encoding Form (CEF). Thus, CEF specifies how exactly the code points will be stored in the memory; whether they will be stored in a sequence of memory units of length 8 bits, or of length 16 bits or of length 32 bits. Unicode defines three Character Encoding Forms: UTF-8 which maps the code points into a sequence of code units, each code unit being 8 bits long, UTF-16 which maps the code points into a sequence of code units, each code unit being 16 bits long and UTF-32 in which each code unit is 32 bits long.

The sequence of code units do not necessarily have the same length for every code point of the Coded Character Set. For a fixed-width encoding, all code unit sequences have the same length while in variable-width encoding the length of the sequences vary from code point to code point.

Once the code unit sequences are obtained, Unicode maps them into serialised byte streams. This mapping is known as the Character Encoding Scheme (CES). The CES eliminates the need of byte-swapping for cross-platform data, which would otherwise be required for all the code unit sequences having length greater than 8 bits. The reason for this is that the CES converts the code unit sequences in either big-endian order or little-endian order.

In addition to this, Unicode also defines the transfer encoding syntax which is transformation that may be performed on serialised byte streams so as to optimise it for some situations, for example, transforming a sequence of 8-bit values for transmission through a system that handles only 7-bit values.

Having discussed the encoding process employed by Unicode let us next have a look at the three popular Character Encoding Forms: UTF-8, UTF-16 and UTF-32 and the transfer encoding syntax UTF-7.

5.3.6.1. UTF-32

This is the simplest possible form of Unicode encoding. In this, 32-bits are used to store the code points of all the possible characters and hence the name UTF-32. The problem with this encoding is that for documents for which an ASCII-like encoding will suffice, UTF-32 uses four times the space used by ASCII-like encodings and hence is rarely used in practice.

5.3.6.2. UTF-16

This is a variable width encoding which uses either one or two 16-bit words for each character. The byte order within a word is hardware dependent and hence all UTF-16 data streams are required to use the character U+FEFF or U+FFFE at the beginning of the stream to indicate the byte order used. This is not considered as a part of the data but is only the byte order marker.

5.3.6.3. UTF-8

This encoding system stores the code points into a series of one or more 8-bit bytes. It is a variable-width encoding in which the first 128 code-points are represented by one byte and are similar to the ASCII encoding. The remaining code-points use two to six bytes. This encoding has several advantages:

- No changes have to be done to support ASCII encoded data stream - files and strings containing only 7-bit ASCII characters have the same encoding in both UTF-8 and ASCII.
- Most functions of the standard library of the C programming language, that have been used for character processing, work because they operate on 8-bit values.
- For texts that use relatively less number of non-ASCII characters this encoding proves to be very space efficient because it will require only slightly more than 1 byte per character, on an average.

This encoding form is the way for using Unicode under Unix-style operating systm. It is recommended to use UTF-8 as your encoding. to know more about UTF-8 please refer [ENT12].

5.3.6.4. UTF-7

This is very similar to UTF-8 except that the high-bit is always zero. This is useful because MIME, often used in email systems, requires that the encoding used to send emails be 7-bit ASCII. So if any email uses Unicode encoding it is invalid. UTF-7 allows mail to use Unicode and also confine to standards. Any standard ASCII character is encoded as is and characters with code points above 128 is encoded using an escape

sequence or a '+' character followed by the Unicode encoded character encoded in Base64, and terminated by a '-'. Literal '+' characters are encoded with '+-'.

5.3.7. UCS 2 and UCS 4

UCS-2 and UCS-4 are defined by ISO 10646-1. These are sequences of 2 bytes and 4 bytes per character respectively. UCS-4 can represent all the UCS and Unicode characters while UCS-2 can represent only those characters of the BMP that fall in the range U+0000 and U+FFFF.

5.3.8. Difference between ISO 10646 and Unicode

The Unicode Standard published by the Unicode Consortium corresponds to ISO 10646 at implementation level 3. All the characters in both the levels have the same names and code values. But in addition to this, the Unicode Standard defines more semantics associated with some characters. Unicode also specifies algorithms for rendering presentation forms of some scripts like Arabic, handling bi-directional texts and algorithms for sorting and string comparisons.

ISO 10646, on the other hand, is much simpler. It specifies some terminology related to the standard, defines some encoding alternatives and it contains specifications of how to use UCS in connection with other established ISO standards.

5.4. Normalisation

Text in computers can be encoded in one of many character encodings as we saw in previous sections. Some encoding schemes allow multiple representations of the 'same' string. There are a number of operations that are sensitive to the multiple representations of the same character like string matching, indexing, searching, sorting etc. String matching is usually performed by comparing two strings byte by byte, which does not work if the strings are encoded in different encodings or they are encoded in the same encoding but shows variations allowed for the 'same' string by the use of combining characters. For example, ã can be written as a + ~ or a - ~. There are also cases where the order of combining characters does not matter. For example, the pair of characters ¸´ can be written with either one first. A direct byte-by-byte matching won't work in such a case. Incorrect string matching can have far reaching consequences, including creation of security holes. Solving the string matching problem involves Normalisation, which in a nutshell means bringing two strings which are to be compared to a common, canonical encoding before performing binary matching on them.

Normalisation can be divided into two categories depending on of when it is performed. Normalisation can be performed when strings are created, called early normalisation or when the strings are compared, called late normalisation.

In response to the need for a normalised form, the Unicode Consortium provides four standard normalisation forms - Normalisation Form D (NFD), Normalisation form C

(NFC), Normalisation form KD (NFKD) and Normalisation form KC (NFKC). For more information on the same see UTR#15: Unicode Normalisation Forms (http://www.unicode.org/unicode/reports/tr15/) [ENT11]. Normalisation Form C is designed to favour precomposed characters such as ã over combining character sequences a and ~

5.5. Collation

Collation refers to the process of ordering (arranging) characters. Sorting string is a common action but is not as trivial as it may sound. This is due to the fact that not everyone uses the same set of characters and not everyone uses the characters in the same way. A collation is a defined way of sorting strings and is often language dependent. The Unicode Collation Algorithm is a general purpose algorithm for sorting Unicode strings.

5.6. Encoding and Localisation

Encoding is very important to localisation since input and output of text in any language requires encoding information. The application should specify what encoding it uses so that the strings are stored correctly without loss of data.

String processing and display also requires knowledge of the encoding used for the string. If a string uses a certain encoding scheme, say UTF-8 and the rendering engine that will render this string is not aware that it uses UTF-8 and applies say, ASCII encoding to render it, then all the content will appear gibberish. This will be a major hindrance to sell localised applications. Hence it is very important to specify what encoding a string uses in order to interpret it and display it correctly to the users. There are standard ways of doing this.

For example, for an email message you can find a string of the form

Content-Type: text/plain;charset="UTF-8"

in the header.

For a web page you can specify a meta tag in the head section of the HTML page as follows:

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

However, care should be taken so that this tag is the first in the head section. This is because as soon as the web browser sees this tag it will stop parsing the page and start fresh interpreting the whole page using the encoding specified.

5.7. Suggestions

It is suggested that you use Unicode as your encoding. The reasons for this include:

- This is the only universal Character repertoire available today.
- It is being updated/completed carefully
- It is widely accepted and implemented by the industry

Chapter 6. Locale

6.1. Introduction

In software terms, locale refers to a collection of information associated with a country or region. This collection of information includes the language spoken in the region, date format, number format, currency format, measurement units, scripts and local names for timezones.

The concept of locale was introduced with internationalisation (i18n) in which generic frameworks are made so that software programs can easily adjust themselves to different language and cultural conventions.

Users can configure their system to pick up a locale that suits them. The software programs then load the locale definition files for their working. Therefore, in order to create a locale one needs to create a locale definition file, where it will put all the information specific to that locale will be kept. Don't know about locale definition file? see Section 6.6 for Locale Definition File. The locale definition file helps users to get make internationalised software support a new language and culture without actually modifying the software.

Locale information also includes a name and identifier of the language spoken. Identifier is used to retrieve information about the locale. This identifier is a 32-bit unsigned integer that is divided into four parts. The first two parts specify the language and the last two parts specify the sorting order of text strings. Thus locale not only decides which character set to use or what convention to use for date, currency etc. but also specifies the sort order to use.

6.2. Locale Naming

Read if: you have no knowledge about how to name a locale.

A locale is identified by the language, country and character set. The naming convention for a locale as specified by open i18n guideline is:

{lang}-{territory}-{codeset}[@{modifier}], where

{lang} is the 2 letter language code defined in ISO 639:1988. In the absence of a valid 2-letter code, the 3-letter language code as defined in ISO 639-2[TLC4] can be used. You can find the specific language codes in ISO 639-2 Registration Authority at Library of Congress .

{territory} is the 2-letter country code defined in ISO 3166-1:1997. You can get these codes from ISO 3166 Maintenance agencyi[TLC5] .

{codeset} describes the character set used in the locale.

{modifiers} This part is optional. This adds more information to the locales. This can be done by setting options (turn on flags or use "=" to set values). Options are separated by commas.

e.g. If "hl" is the 2-letter language code for "Hypolan" used in the country "Hyporeg" which has country-code "HR" then the locale name hl_HR. ISO8859-1 indicates a locale based on hypolan language of Hyporeg region using ISO-8859-1 character set. Thus, for "Hindi" ("hi") spoken in "India" ("IN") the name of the locale used is hi_IN.ISCII-DEV, where ISCII-DEV is the encoding. Similarly, fr_FR ISO8859-1 is the name of the locale for the "French" language spoken in "France" and fr_CH.ISO8859-1 is the name of the locale for the "French" language spoke in "Switzerland" ("CH"). The encoding used in both these locales is "ISO-8859-1".

6.3. Locale Management

Read if: you are not aware of how to find your current locale setting / how to build a locale / how to override parts of a locale / how to set a locale.

6.3.1. Finding out your current locale

To know the current locale setting in your Linux system use the locale command. The locale command gives information about the current locale environment or all locales. When invoked without arguments, the locale command details the current locale environment for all the locale categories. You can see the same in the figures below.

Figure 6-1. Finding the current setting of the 'LC_TIME' variable

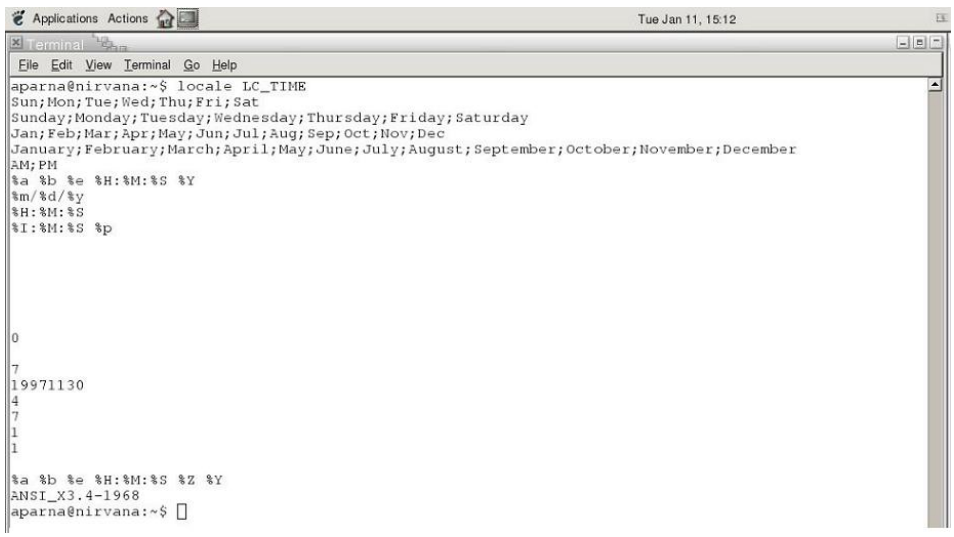


Figure 6-2. Finding the current setting of the 'LC_CTYPE' variable

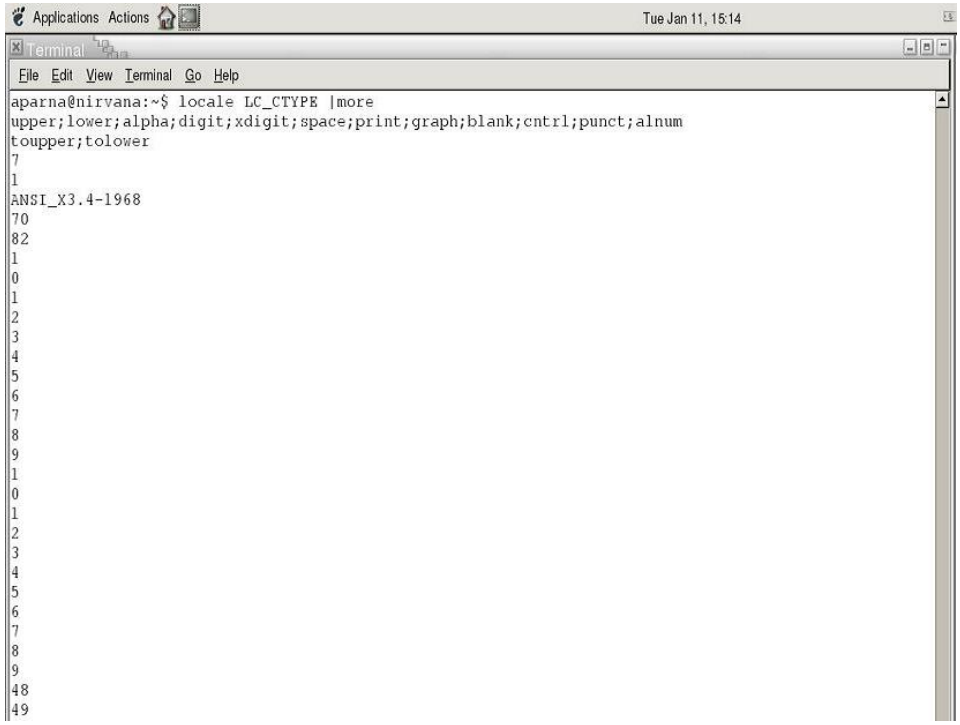
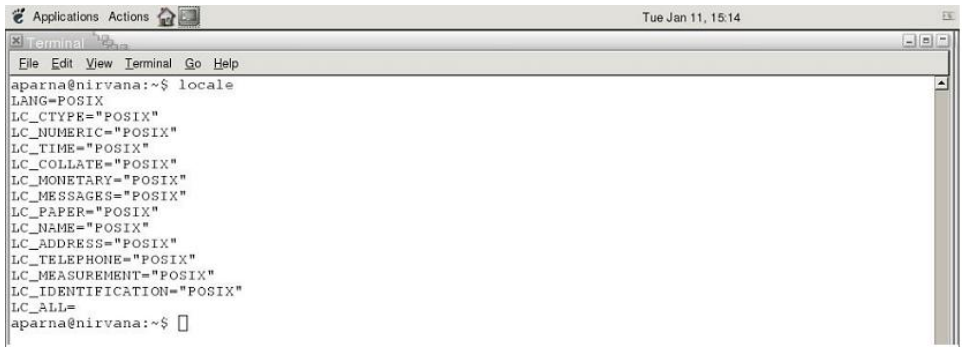


Figure 6-3. Running the locale command without any argument



6.3.2. Building a locale

In order to build a locale you need the following things:

- A locale definition file in which you will describe the data for the various locale categories. See Section 6.6 for details of Locale Definition File.
- If you are defining your own character set then you will need to create a charmap file for it giving every character a symbolic name and describing the encoded byte strings. Don't know what a charmap file is? See Section 6.5.

Once you have the locale definition file and the charmap file, you will need to compile them to create a locale. You use the localedef command to compile the files. The syntax of the 'localedef' is localedef [-f {charmap}] [-i {input} {name}] . For example, if you have to build a locale hi_IN1 from hi_IN locale definition file and using charmap UTF-8, then your command will be localedef -f UTF-8 -i hi_IN hi_IN1. Please note that you need root access to create the locale.

6.3.3. Setting a locale and over-riding parts of a locale

You can set your own by assigning a locale name to the LANG environment variable. You can also assign values to one of the below mentioned environment variables mentioned below. These environment variables influence specific locale functions as described below:

- LC_COLLATE: This variable specifies the collating sequence to use when sorting strings.
- LC_CTYPE: This variable defines the character set to be used by your locale.
- LC_MONETARY: This variable defines the monetary formats.
- LC_NUMERIC: The number formats are specified by this variable.
- LC_MESSAGES: This variable can be used for specifying the language in which messages will appear if translations are available and also the strings for affirmative and negative responses.
- LC_TIME: This variable specifies the date and time format.
- LC_ALL: This overrides all the other variables including the LANG variable.

You can assign values to each of these variables in the same way as you do to the LANG variable; assign locale names to them. Thus if you want to override only some parts of the locale all you need to do is assign the appropriate values to the corresponding environment variables.

6.4. Submitting a locale

Once you create a locale you should submit the same for inclusion in glibc so that others can use your locale. You can also add patches to the locales listed in glibc. You can submit your new locale or modified locale as a Request For Enhancement (RFE) to the sourceware bugzilla [TLC8]. You will have to first get an account there by registering for the same. Once you create an account you can login and go here (http://sources.redhat.com/bugzilla/enter_bug.cgi) [TLC9]. You will get a page which you will have to fill as follows:

- version : Unspecified priority: P2

- component: locale data
- severity: enhancement
- assigned to & CC: leave blank
- Host triplet : the version of glibc you are using (e.g. 2.3.3)
- Target triplet & Build triplet: leave blank
- Summary: "RFE [name of your locale] Locale Data"
- Description: paste the text of your locale file in the Description field.

Once you have done the above press the commit button.

It is always good to get the original locale creator involved when submitting a change request. This increases the chance of the glibc maintainers approving the change request. More information on the list of authors and contributors to the locale and how-to submit a locale can be found at [TLC6] and [TLC7].

6.5. Character Set Description (charmap) Source File Format

The charmap source file defines character symbols as character encodings - it defines the encodings for the characters in the Portable Character Set (http://www.sensi.org/~alec/man/man_b/p_charset.html) [ENT14] and may define the encoding for the additional characters supported by the implementation. Thus, this is the place where you can specify the encoding that your locale would use. It also assigns standardised names to the characters in the portable character set, thus making it possible to refer to any character of the set regardless of the encoding.

Each symbolic name specified in the Portable Character Set (http://www.sensi.org/~alec/man/man_b/p_charset.html) [ENT14] is mapped to a unique encoding value. Below is a list of declarations that can precede the character definitions. These declarations specify the characters to be used as comment or escape character, maximum and minimum number of bytes in a character and the name of the coded character set.

- <code_set_name> - This specifies the name of the coded character set for which the charmap file is defined.
- <mb_cur_max> - This specifies the maximum number of bytes in a multi-byte character. The default is 1.
- <mb_cur_min> - This is an unsigned positive integer value which specifies the minimum number of bytes in a character for the encoded character set.

- <escape_char> - This defines the character to be used as the escape character. The characters following the escape character will be treated in a special way. The default escape character is '\'.
- <comment_char> - This defines the the character which will be used to mark a line as comment. The comment character when placed on the column 1 of any line will make the entire line to be treated as comment. The default is '#'.

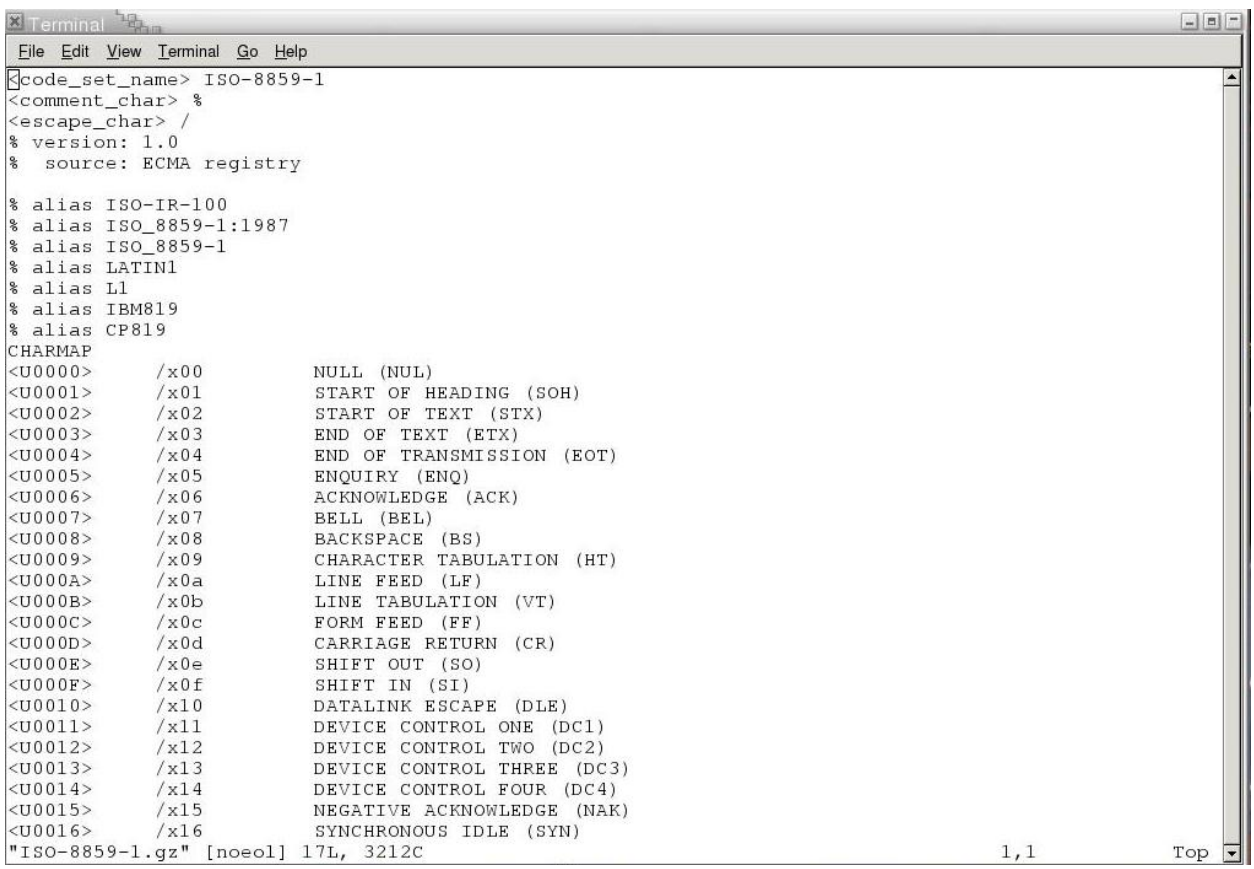
Please note that the above keywords should be specified as is (along with the brackets) in the definition.

All the lines immediately following the CHARMAP header line and preceding the END CHARMAP trailer line will be the character set mapping definitions. Empty lines and lines with the comment character in the first column will be ignored. Each non-empty, non-comment line should be in one of the following forms:

- <symbolic-name>,<encoding>,<comments>
- <symbolic-name>,<symbolic-name>,<encoding>,<comments> - In this you can give a range of one or more symbolic names.

The following is figure is the screenshot of a charmap file.

Figure 6-4. Charmap file



6.6. Locale Definition File

The locale definition file contains all the information required by the localedef command to convert it into the binary locale database. The file starts with a header that may consist of the following keywords:

<escape_char> followed by the character which should be used as the escape character in the rest of the file.

<comment_char> followed by the character which should be used as a comment character. The default is '#'.

The locale definition file has one section for each locale category. The various possible locale categories and the associated options are explained in the next section. Each category definition consists of:

- a category header which is essentially the category name.
- The associated keyword/value pairs that comprise the section body.
- The category trailer which is essentially 'END <category_name>' . For example,

```
LC_CTYPE
  source for LC_CTYPE category.
END LC_CTYPE
```

Each of the category sections can be copied from another existing locale or can be defined from scratch. If you wish to copy the category from another locale you can do so using the copy keyword followed by the locale which should be copied. Please note that if the copy keyword is used, then no other keyword can be specified in the definition. Hence if you want to copy a particular category from some other locale you will have to do it totally. Partial copy is not possible.

6.7. Locale Categories

Read if: you are not aware of the different locale categories or how to define a locale category for your locale definition file.

The various locale categories which are defined in the locale definition file are as follows:

6.7.1. LC_CTYPE

This category specifies which characters are considered as alphabetic, numeric, punctuation, hexadecimal, blank, control characters and so on in the chosen locale. It also defines case conversions.

This category begins with a LC_CTYPE category header and ends with an END LC_CTYPE category trailer. All the operands of this category statements are defined as a list of characters. Each list contains one or more characters or symbolic character names separated by semicolon.

The following keywords are recognised in the LC_CTYPE category. The characters specified with the keywords will be provided if they are missing and will be accepted if they are present.

- copy: Specifies the name of the existing locale from which the definition of this category has to be copied. If this keyword is used, then no other keyword can be used within LC_CTYPE.
- upper: All characters which should be considered as uppercase letter characters are defined by this keyword.The characters defined by the cntrl, digit, punct or space cannot be specified with this keyword. At a minimum, the letters from A-Z must be defined.
- lower: All characters to be included as lowercase letter characters are defined by this keyword. The characters defined by the cntrl, digit, punct or space cannot be specified with this keyword. At a minimum, the letters from a-z must be defined.
- alpha: Defines all the letter characters. The characters defined by the cntrl, digit, punct or space cannot be specified with this keyword. The characters defined in upper and lower are automatically included in this group.
- digit: This set defines all the digit characters.
- alnum: Defines all the alphanumeric characters. The characters defined alpha and digit are included automatically. The characters defined by the cntrl, punct or space cannot be specified with this keyword.
- space: Defines whitespace characters. The characters defined by the cntrl, alpha, upper, lower, digit, graph or xdigit cannot be specified with this keyword. At a minimum, the whitespace characters <space>, <form-feed>, <tab>, <newline>, <carriage-return> and <vertical tab> should be specified. Also the characters specified with the blank keyword must also be specified.
- cntrl: Defines the control characters. No character defined by the alpha, upper, lower, digit, graph, punct, print, space or xdigit can be specified.
- punct: Defines punctuation characters. Characters defined by alpha, upper, lower, digit, space or xdigit cannot be specified.

- graph: Defines printable characters excluding the <space> character. If you do not define characters with this keyword then all characters defined by the alpha, upper, lower, digit, xdigit and punct are automatically included in this set. The characters defined by the cntrl keyword cannot be specified in this list.
- print: Defines printable characters including the <space> character. If you do not define characters with this keyword then all characters defined by the alpha, upper, lower, digit, xdigit and punct are automatically included in this set. The characters defined by the cntrl keyword cannot be specified in this list.
- xdigit: Defines hexadecimal digit characters. This class defaults to its normal class limits.
- blank: Defines blank characters. The characters <space> and <horizontal-tab> are included in this group if this keyword is not specified. All the characters in this list are automatically included in the list defined by the keyword space.
- charclass: Defines one or more character class names as strings separated by semicolons. Each named character class can then be defined subsequently in the LC_CTYPE definition.
- toupper: Defines the mapping of lower-case characters to uppercase characters. Operands to this keywords consists of semicolon separated character pairs. Each character pair is enclosed within () (parentheses) and the first pair is separated from the second by comma.
- tolower: Defines mapping of uppercase characters to lowercase characters. Operands to this keyword are defined as they are done for the toupper keyword.

The following is an example of LC_CTYPE definition

```
LC_CTYPE
#"alpha" is by default "upper" and "lower"
#"alnum" is by default "alpha" and "digit"
#"print" is by default "alnum", "punct" and the space character
#"graph" is by default "alnum" and "punct"
#"tolower" is by default the reverse mapping of "toupper"

#
upper           <A>;<B>;<C>;<D>;<E>;<F>;<G>;<H>;<I>;<J>;<K>;<L>;<M>;\
                <N>;<O>;<P>;<Q>;<R>;<S>;<T>;<U>;<V>;<W>;<X>;<Y>;<Z>
#
lower           <a>;<b>;<c>;<d>;<e>;<f>;<g>;<h>;<i>;<j>;<k>;<l>;<m>;\
                <n>;<o>;<p>;<q>;<r>;<s>;<t>;<u>;<v>;<w>;<x>;<y>;<z>
#
digit           <zero>;<one>;<two>;<three>;<four>;<five>;<six>;\
                <seven>;<eight>;<nine>
#
space           <tab>;<newline>;<vertical-tab>;<form-feed>;\
```

```
                <carriage-return>;<space>
#
cntrl           <alert>;<backspace>;<tab>;<newline>;<vertical-tab>;/
                <form-feed>;<carriage-return>;<NUL>;<SOH>;<STX>;/
                <ETX>;<EOT>;<ENQ>;<ACK>;<SO>;<SI>;<DLE>;<DC1>;<DC2>;/
                <DC3>;<DC4>;<NAK>;<SYN>;<ETB>;<CAN>;<EM>;<SUB>;/
             <ESC>;<IS4>;<IS3>;<IS2>;<IS1>;<DEL>
#
punct           <exclamation-mark>;<quotation-mark>;<number-sign>;\
                <dollar-sign>;<percent-sign>;<ampersand>;<asterisk>;\
                <apostrophe>;<left-parenthesis>;<right-parenthesis>;
                <plus-sign>;<comma>;<hyphen>;<period>;<slash>;/
                <colon>;<semicolon>;<less-than-sign>;<equals-sign>;\
                <greater-than-sign>;<question-mark>;<commercial-at>;\
                <left-square-bracket>;<backslash>;<circumflex>;\
                <right-square-bracket>;<underline>;<grave-accent>;\
                <left-curly-bracket>;<vertical-line>;<tilde>;\
                        <right-curly-bracket>
#
xdigit          <zero>;<one>;<two>;<three>;<four>;<five>;<six>;\
                <seven>;<eight>;<nine>;<A>;<B>;<C>;<D>;<E>;<F>;\
                        <a>;<b>;<c>;<d>;<e>;<f>
#
blank           <space>;<tab>
        #
toupper  (<a>,<A>);(<b>,<B>);(<c>,<C>);(<d>,<D>);(<e>,<E>);\
               (<f>,<F>);(<g>,<G>);(<h>,<H>);(<i>,<I>);(<j>,<J>);\
               (<k>,<K>);(<l>,<L>);(<m>,<M>);(<n>,<N>);(<o>,<O>);\
                (<p>,<P>);(<q>,<Q>);(<r>,<R>);(<s>,<S>);(<t>,<T>);\
               (<u>,<U>);(<v>,<V>);(<w>,<W>);(<x>,<X>);(<y>,<Y>);\
               (<z>,<Z>)
       #
       END LC_CTYPE
```

6.7.2. LC_NUMERIC

This category defines rules and formatting characters and rules for non-monetary numeric information. This category section begins with the category header LC_NUMERIC and ends with the category trailer END LC_NUMERIC.

In this category, the operands to the keywords are either string or integer values. String operands are enclosed within " " (double quotation marks). The operands and keywords are separated by one or more spaces. Two adjacent double quotation marks indicate undefined string values and a -1 indicates an undefined integer value.

The following is the list of keywords used in this category and a description of what they stand for:

- copy: This statement specifies the name of the locale from which this category's description has to be copied. If this keyword is used, then no other keyword can be used in this section.
- decimal_point: This defines the string used as the decimal delimiter for numeric, non-monetary quantities.
- thousands_sep: This defines the string separator used for grouping digits to the left of the decimal delimiter.
- grouping: This specifies the size of the groups to the left of the decimal position. The operands to this keyword is a set of integers separated by semicolon. The first integer indicates the length of the group immediately to the left of the decimal point. The following integers indicate the length of the groups to the left of the previous group. A -1 indicates that no more groups are possible. If -1 is not the last integer in the set of operands then the size of the previous group (if any) will be repeatedly used for the remainder of the digits.

  The following is an example. Suppose 123456789 is the value to be formatted and the thousands_sep is ' then the grouping results for the following values are:

  Table 6-1. Grouping

Grouping value	Formatted value
3;-1	123456'789
3;4;-1	12'3456'789
4;2	1'23'45'6789

6.7.3. LC_MESSAGES

This section defines the format of affirmative and negative system responses. This category definition begins with LC_MESSAGES and ends with END LC_MESSAGES. The operands in this category are defined as strings or regular expressions enclosed in " ". Two adjacent " " (double-quotation marks) indicate an undefined value. The following keywords are recognised in this category :

- copy: Specifies the name of the locale whose LC_MESSAGES definition has to be used. If this keyword is specified then no other keyword can be specified.
- yesexpr: Specifies an extended regular expression that describes the acceptable affirmative responses for a question expecting an affirmative or negative response.

- noexpr: Specifies an extended regular expression that describes the acceptable negative responses for a question expecting an affirmative or negative response.

6.7.4. LC_TIME

This category defines rules for formatting the date and time information. This category section begins with the category header LC_TIME and ends with the category trailer END LC_TIME.

In this category, the operands to the keywords are either string or integer values. String operands are enclosed within " " (double quotation marks). The operands and keywords are separated by one or more spaces. Two adjacent double quotation marks indicate undefined string values and a -1 indicates an undefined integer value.

The following is the list of keywords used in this category and a description of what they stand for:

- copy: (as explained in previous categories)
- abday: This keyword is followed by a list of seven abbreviated weekday names for the seven days of the week. The list starts with Sunday and the abbreviated strings are separated by a semi-colon. Each string must be of equal length and must contain five characters or less. The strings correspond to the %a field descriptor. To know more about field descriptors please refer to [TLC10].
- day: This keyword is followed by a list of strings that define the full spelling of the weekday names. The list consists of seven semi-colon separated strings with the first string corresponding to Sunday or its translation. The names correspond to the %A field descriptor. To know more about field descriptors please refer to [TLC10].
- abmon: This keyword is followed by a list of strings that define the abbreviated month names. The list consists of 12 semi-colon separated values. Each string must be of equal length and contain 5 characters or less. The first string in the list corresponds to the first month of the year, the second string corresponds to the second month and so on. The names correspond to the %b field descriptor. To know more about field descriptors please refer to [TLC10].
- mon: This keyword is followed by a list of strings that define the full spelling of the month names. The list consists of 12 semi-colon separated strings with the first string corresponding to the first month of the year. The names correspond to the %B field descriptor. To know more about field descriptors please refer to [TLC10].
- am_pm: This keyword is followed by a list of strings that are used to represent the ante meridiem and post meridiem values. These values correspond to the %p field descriptors. To know more about field descriptors please refer to [TLC10]. The list consists of 2 semi-colon separated strings with the first string corresponding to the ante meridiem and the second to the post meridiem values.

- d_t_fmt: This keyword is followed by a string that is used to represent the standard date and time formats. These values correspond to the %c field descriptor. The string can contain a combination of characters and field descriptors. To know more about field descriptors please refer to [TLC10].
- d_fmt: This keyword is followed by a string that is used to represent the standard date format. This value corresponds to the %x field descriptor. The string can contain a combination of characters and field descriptors. To know more about field descriptors please refer to [TLC10].
- t_fmt: This keyword is followed by a string that is used to represent the standard time format. This value correspond to the %x field descriptor. The string can contain a combination of characters and field descriptors. To know more about field descriptors please refer to [TLC10].
- t_fmt_ampm_time: This keyword is followed by a string that is used for the standard 12-hour clock format that includes the am_pm value. The value of this string correspond to the %r field descriptor. The string can contain a combination of characters and field descriptors. To know more about field descriptors please refer to [TLC10].
- era: Defines how the years are counted and displayed for each era in a locale. For more information on how to define era and for information on the various field-descriptors used in his category referto [TLC10].

6.7.5. LC_COLLATE

This category defines the collating information for characters defined in the charmap file (or the portable character set in case the charmap file is not defined). For defining any collating information you need to define a collating element. A collating element is a unit of comparison for collation. Collating elements can be a single character or a sequence of characters. Every collating element has weights attached to it. These weights determine if the collating element collates before, equal-to or after other collating elements. The number of collation weights for any collating element is equal to the number of collation orders defined for the locale. You can define multiple collation orders in your locale. For example, the French locale defines two collation orders, one with a primary set of weights and the other with secondary of weights. If two collating elements' primary set of weights equate then the secondary set of weights are considered for collation. The localedef command assigns weights to each collating element.

Every character in the charmap file (or portable character set,in case the charmap file is not defined) is a collating element. You can define additional collating elements in your locale using the collating-element statement. The syntax of this is:

collating-element character-symbol from string, where character-symbol value defines the collating element that is a string of one or more characters and string is a string of two or more values that define the collating-symbol value. Following are the examples

of defining collating elements:

- collating-element <ch> from <c><h>
- collating-element <f1> from <f><1>

This category begins with a LC_COLLATE header and ends with END LC_COLLATE trailer. The following keywords are recognised in this category:

- copy: (purpose same as in the other categories)
- collating-element: (explained above)
- collating-symbol and order_start - (For explanation of these keywords please refer to [TLC11]

6.7.6. LC_MONETARY

This category defines the rules and symbols for formatting monetary information. This category begins with LC_MONETARY header and ends with END LC_MONETARY trailer. All operands in this category are defined as strings or integer values. Strings are enclosed in " " and two adjacent "" indicate and undefined value. All the operands are separated from the keywords by one space. A -1 indicates an undefined integer value. LC_MONETARY has many keywords. You can find the keywords and examples at [TLC12].

Chapter 7. Fonts

7.1. What is a Font?

Read if: A general understanding of fonts is recommended for those deeply associated with Localisation.

A font is a visual representation of characters from a particular character set. The font for a character normally consists of a set of images, called glyphs. A glyph is the visual representation of a character or a part of it. The mapping between glyphs and characters need not be one to one. A glyph may contain many characters (called a ligature, for example, fi) and basing on the context, a character may have more than one glyph and a glyph may be part of many characters.

Figure 7-1. Glyph: Visual representation of a character

Figure 7-2. A character is a combination of glyphs

The character (right side of the arrow) in the above diagram is the combination of other three glyphs(left side of the arrow)

Figure 7-3. A glyph having more than one character, aka ligature[1]

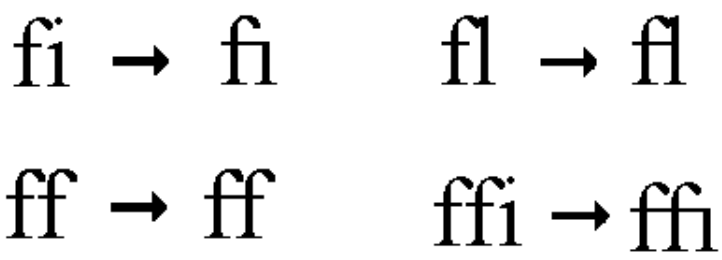
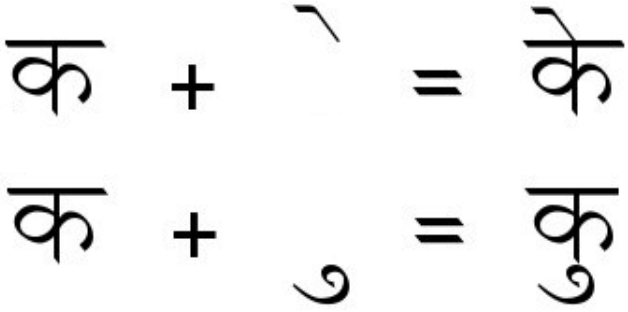


Figure 7-4. A glyph becoming part of more than one character



Based on various properties, (such as typeface etc) fonts can be classified along many dimensions.

- Fixed width Vs Variable width [2]
- Serif Vs Sanserif [3]

In general information associated with a font is captured with one or two files depending upon the type of the font. How ever this information includes the following

- Font Metric

  Font metrics define the spacing between variable width fonts. It include information like size of the font, kerning[4] information etc.

- Font Shape

This is the outline or shape of the font. The components of font shape is a stroke, an accent etc. The font shape is defined in terms of the glyphs for that particular character set.

- Typeface

  Typeface is the style or design of the font. For example bold, italic, and bold-italic are examples of typefaces.

Fonts is a fascinating and rich field of study involving many aspects such as display medium, purpose, etc. For our requirement, a limited understanding covering the above aspects would suffice; hence we limit our discussion of fonts to these. Interested readers can refer to the resources given at the end for a more detailed study.

7.2. Types of Fonts

Depending upon the way Fonts are stored/represented, Fonts can be categorised as follows. Today, mostly true type and open type fonts are used.

7.2.1. Bitmap Fonts

A bitmap is nothing but a matrix of dots. Each character or glyph in a Bitmap font is stored as an array of dots (pixels). Bitmap fonts are very easy to generate and do not require complex rendering. Bitmap fonts are good as long as the characters to be rendered are of fixed sizes. They will look blurred when scaled. Bitmap fonts are widely used under X Window System for applications like terminal windows, consoles and text editors where scaling is not required. Depending upon the type of the output used, there are two types of bitmap fonts, namely Bitmap printer fonts and Bitmap screen fonts. Screen fonts are distributed in Bitmap Description/Display Format(BDF).

7.2.2. Outline or Vector Fonts

A character or a glyph in an outline font is described as a set of lines and curves. These fonts can be scaled to any size. Outline fonts can be hinted[5] at smaller sizes and anti-aliased[6] at larger sizes for better results. As opposed to bitmap fonts, outline fonts need sophisticated rendering technology; as a result they produce better display quality compared to bitmap fonts. The following are the various types of outline fonts based on the type of outlines they use.

7.2.2.1. Type1 Fonts

Type1 fonts, devised by Adobe, are widely used scalable fonts under Linux. They are supported by Adobe's Postscript standard. Type1 format is used for printing in Linux, as it is supported by Ghostscript (Postscript interpreter). A Type1 font contains two files, namely metric file and outline file. The outline file contains the descriptions of the character shapes, i.e., stroke, accent etc. It does not contain enough information for typesetting with the font. Outline files comes in two formats, namely PostScript Font ASCII[.pfa] and PostScript Font Binary[.pfb]. The metric file contains data such as Kerning information, Ligature information, spacing between the variable width fonts etc. Metric files are distributed in Adobe Font Metric[.afm] format. In Microsoft Windows they are distributed in .pfm format. The disadvantage of Type1 fonts is that, the metrics associated with them are not enough for complex scripts like Arabic, Indic, Thai etc.

More information on Type1 fonts is available at
http://www.math.utah.edu/~beebe/fonts/postscript-type-1-fonts.html

7.2.2.2. TrueType Fonts

TrueType font format was developed by Apple and later extended by Microsoft. TrueType is widely supported over the web. TrueType font rendering is well hinted and good for screen display. TrueType fonts are distributed in a single file (.ttf extension) which contains shape as well as metric information. These are supported by GNU/Linux, Microsoft Windows, and Apple Macintosh, and all BSD variants(as they use Xfree86 or X.org). Recently FreeType library has been developed for rendering TrueType fonts under X, which made them available to Linux. They are now becoming popular under GNU/Linux. But there is a possibility that the hinting of TrueType fonts may be covered by a patent from Apple. As a solution, FreeType community is creating new rendering engines, that use other techniques for hinting.

More information on TrueType fonts is available at Microsoft Typography page (http://www.microsoft.com/typography/WhatIsTrueType.mspx) [FOT4].

7.2.2.3. OpenType Fonts

OpenType format is a combined initiative of Adobe and Microsoft with the aim to create an advanced font format required for internationalisation. The basic structure is same as TrueType fonts except that it can take either a Type1 or TrueType outline. OpenType fonts are distributed as a single file (.otf extension). The FreeType2 library on GNU/Linux has some support for OpenType fonts, so they can be used under Linux. As you can keep complex rendering information with an OpenType font, this is the best font format available and a definite choice for localisation.

More information on OpenType fonts is available at Adobe Typography page (http://store.adobe.com/type/opentype/main.html) [OTF1].

7.2.2.4. Comparison

TrueType fonts has better support for hinting than the Type1.

Type1 fonts uses cubic curves for glyphs as opposed to quadratic curves in the case of TrueType. Quadratic curves are less powerful than cubic curves. There are very limited tools that provide editing of quadratic curves.

TrueType fonts are widely used because of its association with Windows.

And from the localisation point of view, Type1 metric file can't contain necessary information to handle complex scripts like Arabic, Indic, and Thai. So TrueType fonts are better for localisation purposes compared to Type1 fonts. But OpenType fonts are the preferred fonts for Localisation purpose because of their superiority to other formats.

7.3. Fonts and GNU/Linux

Read if: You want to know the way GNU/Linux handles fonts

Traditionally, in a Linux system, X Server[XFree86] is responsible for serving the font requests of various applications. XFree86 includes two independent font systems, namely 'Core X11 font system' and 'X FreeType Interface library font system'(Xft).

The X11 core font system is present in all the distributions of X11, but could only deal with bitmap fonts in the earlier versions. Over the years it has been extended to support scalable fonts like Type1, and now TrueType, OpenType etc.

The Xft font system, a relatively recent implementation, was designed to provide good and efficient support for scalable fonts. It supports advanced features such as anti-aliasing, sub-pixel rendering etc. It also allows applications to use fonts that are not installed system wide for displaying documents with embedded fonts. But not all applications use Xft, as usage of Xft requires making extensive changes to toolkits (ex: Qt, GTK, XUL etc) . XFree86 is currently continuing to maintain the core font system. The toolkit authors are slowly switching to Xft, and they are being encouraged to do so. As of now GNOME and KDE, the popular Desktop environments in GNU/Linux, use Xft by default.

To serve the fonts over the network, X Server also make use of font servers.

First we will discuss all these systems briefly and later we will look at installing fonts for these systems.

Please note that, recently an alternative for XFree86 is available, namely X.org. The structure of X.org is same as XFree86 as they follow the same specification. The configuration of X.org is same as XFree86. If you are using X.org as your X server, you can configure it by following the instructions given below for configuring the XFree86.

7.3.1. The X11 Core Fonts System

XFree86 has a font path, which is a list of several directories and/or font servers (explained below) where it searches for fonts. When an application requests a font, it searches all the directories in the font path one at a time until it finds the best match. It selects truetype fonts over bitmap and un-scalable bitmaps over bitmaps. This is done by scanning twice. On the first scan it looks for the exact match and second scan looks for the best match.

Making a font available to X Server includes creating a font directory and adding it to the font path. X Server should be aware of any font server running on the system so that it can delegate the above task to the font server. This can be achieved by adding the details of font server in the X Server's font path. Preferably the font server should be the first entry in the X Server's font path and only if the font server can't serve the request, X Server will continue searching the entire font path for the font.

7.3.2. X Font Server

A font server is a background process which makes fonts available to the X Server. Font servers are required to send fonts to remote displays. They have their own font path, where it looks for the fonts available. You might be wondering how it works as X Server also has its own font path. If a font server is running(either locally or remotely) , that will be included in the X Server's font path by adding 'unix/:7100' to the X Server's font path. As an effect, the fonts available to the X Fonts Server are also available to the X Server.

But recently X Font servers (for example, Xfs and Xfs-TT) are patched with support for rendering TrueType fonts, making it compulsory to use font server for the users of older version of XFree86 (earlier to 4.X) . This is done with the help of FreeType library. But the latest version of XFree86 (4.X) has built in support for TrueType/OpenType fonts with the help of the same FreeType library.

Currently available Font Servers are Xfs and Xfs-TT. Xfs supports both Type1 and TrueType fonts while Xfs-TT supports only TrueType. Even though using X Font Server is not necessary, make a choice keeping the following points in mind.

- Font server is required to serve the fonts through a network.
- Rendering will be faster, at the cost of memory
- Those who are using older versions of XFree86 (earlier to 4.X) are required to use Font servers for rendering TrueType fonts.

7.3.3. X FreeType interface library and Fontconfig

X FreeType interface library[Xft] provides a client side font API for the X applications. The Xft library was written to provide X applications a convenient interface to the FreeType font rasterizer and the X rendering extension. It uses Fontconfig library to select fonts and X protocol to render them. When the X Server is not blessed with X rendering extension, it uses core X protocol to draw client side glyphs, which makes compatible support of client side fonts for all X Servers.

7.3.3.1. Fontconfig

Fontconfig is a library for configuring and customising font access.

Installing a new font is just keeping the font in the '.fonts/' in the user's home directory.

'Fontconfig' can

- automatically discover new fonts when installed;
- perform font name substitution, i.e., fonts aliasing for selecting alternative fonts when specified fonts are missing;
- identify the set of fonts required to completely cover a set of languages;
- efficiently and quickly find the fonts you need among the set of fonts you have installed, while minimising memory usage; and
- be used in concert with the X Render Extension and FreeType to implement high quality, anti-aliased and sub-pixel rendering text on a display.

Note that Fontconfig does not render fonts, this is left to FreeType library. And Fontconfig is not dependent on the XFree86. In Section 7.3.4.2 we will be explaining installation of fonts under Xft font system.

7.3.4. Setting up the system to display fonts

The steps necessary to install new fonts will depend on the font system and the GNU/Linux distribution you are using. The general steps required to install are explained in this section.

7.3.4.1. Installing fonts for the core X font system

Installing fonts for the core X font system includes

- Setting up the font directory and
- Adding it to the server's font path.

7.3.4.1.1. Setting up the font directory

Setting up the font directory includes creating the directory itself, copying the fonts into that, and making it a font directory by creating a font.dir file. If the fonts are scalable, then we have to create a 'fonts.scale' file before creating 'fonts.dir'. Preferably the font directory should be accessible to all users of the system.

7.3.4.1.1.1. Bitmap Fonts

Bitmap fonts are distributed in BDF format. XFree86 can also handle PCF (compiled BDF) and the obsolete SNF formats. PCF fonts are efficient, so it is desirable to convert BDF to PCF format before installing them. The utility 'bdftopcf' is used to convert BDF to the PCF format. The following command results in a .pcf file.

```
$ bdftopcf eenadu.bdf
```

Then you have to compress the pcf file with the help of gzip, which yields a file with the extension .pcf.gz

```
$ gzip eenadu.pcf
```

Now you create a directory, copy the fonts into that and make it a font directory. This can be achieved as follows

```
$ mkdir /usr/local/share/fonts/bitmap/
$ cp *.pcf.gz /usr/local/share/fonts/bitmap/
$ mkfontdir /usr/local/share/fonts/bitmap/
```

'mkfontdir' creates 'fonts.dir' file making the current directory a font directory.

7.3.4.1.1.2. Scalable Fonts (Type1, TrueType, and OpenType fonts)

Installing scalable fonts is very similar to installing bitmap fonts. However 'mkfontdir' can't recognise scalable font files without indexing them. You must index all the font files in to a file called 'font.scale'. This can be done by using the 'mkfontscale' utility.

```
$ mkfontscale /usr/local/share/fonts/truetype/
$ mkfontdir /usr/local/share/fonts/truetype/
```

Now /usr/local/share/fonts/truetype/ directory contains the files, fonts.scale and fonts.dir.

7.3.4.1.2. Setting up the server's font path

After setting up a font directory, adding a font directory to server's font path is as simple as appending the font directory to the list of already available font directories. The following are two ways of setting up the server's font path.

7.3.4.1.2.1. Setting the path for all the users in the system

As the default font path is specified in the X Server's configuration file (i.e., XFree86Config-4 or XFree86Config) , the font path can be directly updated by editing that file. The resulting changes are system wide and permanent.

The XFree86Config-4 has a Files section, which has a set of 'FontPath' entries. Each entry corresponds to a font directory. The font path is the concatenation of all these

entries in the same order in which they appear in the Files section. To add a new directory to the font path you will make a new 'FontPath' entry in the files section. An example XFree86Config-4's Files section looks like this...

```
Section "Files"

FontPath        "unix/:7100"                    # local font server
# if the local font server has problems, we can fall back on these
FontPath        "/usr/lib/X11/fonts/misc"
FontPath        "/usr/lib/X11/fonts/cyrillic"
FontPath        "/usr/lib/X11/fonts/100dpi/:unscaled"
FontPath        "/usr/lib/X11/fonts/75dpi/:unscaled"
FontPath        "/usr/lib/X11/fonts/Type1"
FontPath        "/usr/lib/X11/fonts/CID"
FontPath        "/usr/lib/X11/fonts/Speedo"
FontPath        "/usr/lib/X11/fonts/100dpi"
FontPath        "/usr/lib/X11/fonts/75dpi"

EndSection
```

7.3.4.1.2.2. Setting the path for the current session

The utility 'xset' comes in handy for this, if you don't want to edit the X Server's config file by hand or you don't have permissions to do so.

You can add a directory to the font path by 'xset fp+'

```
$ xset +fp /usr/share/fonts/ttf/
$ xset fp+ /usr/share/fonts/bitmap/
```

fp+ appends to the font path and +fp adds in the beginning of the path. In the same way fp- removes a directory from the end and -fp removes from the beginning.

Once you have added the font directory to the font path, you have to tell the server to rescan the font path to activate the new fonts immediately. This can be done by 'xset fp rehash' . Note that next time when you add new fonts into the same directory, you just have to run 'xset fp rehash' .

```
$ xset fp rehash
```

'xset fp default' will set the font path to the default font path, i.e., font path given in the X Server's configuration file.

Note that by running 'xset', the font path is modified temporarily for the current session. To make the changes permanent, keep the above commands in .xinitrc (or any other file depending on the way you start X) file in your home directory so that whenever you start the X Server these changes will take effect.

7.3.4.1.2.3. Setting up X Font Server's font path

If you are using stand-alone X Font Server, you might want to install fonts through it. So you will have to add the font directory to the X Font Server's font path. X Font server has a font path which is stored in its configuration file(/etc/X11/fs/config for Xfs). It has an entry called 'catalogue' which contains the list of all font directories. You can directly add the new font directory to that list.

```
catalogue = /usr/lib/X11/fonts/misc/,
    /usr/lib/X11/fonts/cyrillic/,
    /usr/lib/X11/fonts/100dpi/:unscaled,
    /usr/lib/X11/fonts/75dpi/:unscaled,
    /usr/lib/X11/fonts/Type1/,
    /usr/lib/X11/fonts/CID,
    /usr/lib/X11/fonts/Speedo/,
    /usr/lib/X11/fonts/100dpi/,
    /usr/lib/X11/fonts/75dpi/
```

Note that all the directories have a comma at the end except the last one. So when you append a new directory, don't put extra comma at the end and make sure to add a comma at the current last line.

The resulting changes are system wide and permanent. Unfortunately, there is no common utility among all the GNU/Linux distributions to update the font server's font path for a particular profile. Redhat has their own utility called 'chkfntconfig'.

7.3.4.2. Installing fonts for the Xft font system

Xft relies on Fontconfig library to configure and customise fonts. The advantage of this is that the font installation is independent of the type of the font.

Fontconfig looks for fonts in a set of well-known directories, including all of the standard XFree86 directories and also a directory called '.fonts/' in the user's home directory. Installing a font for this system is as simple as copying a font file to any of these directories.

```
$ cp eendadu.ttf ~/.fonts
```

Fontconfig will notice the new font at its next opportunity to rebuild the list of fonts. But to update the font path immediately, you can run the command 'fc-cache'

```
$ fc-cache
```

7.3.5. Font Tips

- The order of fonts in the font path are important. The order should be :scalable,unscalable bitmap, bitmap etc. And also 100dpi fonts should come before

75dpi fonts.

- You can check whether XFree86 has X render extension by running 'xdpyinfo | grep RENDER'. If the output is 'RENDER', X includes the rendering extension.
- You can get the current font path by the command 'xset q'
- To list the available fonts for XFree86 run 'xlsfonts'. X Font server fonts can be listed by running 'xslsfonts'.
- To examine the internal details of the font such as details of glyphs, use 'showfont'
- You can enable 'Hinting' support in the freetype library by modifying '/usr/include/freetype2/freetype/config/ftoption.h'. search for:

  ```
  #undef TT_CONFIG_OPTION_BYTECODE_INTERPRETER
  ```

  and change it to:

  ```
  #define TT_CONFIG_OPTION_BYTECODE_INTERPRETER
  ```

7.4. Availability of Fonts

Fonts for various languages are created and made available by various institutions under various terms. In this section, we mention a few general services and fonts for a few languages. While creating font(if you need to), you can start with available fonts in a similar language and extend/modify as required.

Table 7-1. Fonts, language wise...

Language	Font type	Description	URL
Hindi/Malayalam/Tamil/Marathi/Sanskrit	OpenType/TrueType	Fonts from NCST	http://tdil.mit.gov.in/download/op
Telugu	TrueType	Pothana font	http://www.kavya-nandanam.com/dload.htm
Hindi	TrueType	Hindi font from nonGNU	http://savannah.nongnu.org/project
Kannada	TrueType		http://www.baraha.com/html_help
Bengali	TrueType	for Debian GNU/Linux	http://packages.debian.org/testing/ bangla-fonts

Language	Font type	Description	URL
Indic Unicode	TrueType	A collection of OpenType Unicode Indic fonts for Debian GNU/Linux	http://packages.debian.org/testing/indic-fonts
UCS Outline Fonts	OpenType	Fonts for the entire Unicode	http://savannah.nongnu.org/project

Collection of Open Source fonts (http://eyegene.ophthy.med.umich.edu/unicode/fontguide)

- Edward Trager maintains a list of high quality open-source fonts. Please check if your font is listed; if not, notify him on the availability of your fonts.

Microsoft TrueType webfonts (http://freedesktop.org/~fontconfig/webfonts/)

- Microsoft's truetype fonts for English.

GNU Free Bitmap fonts (ftp://ftp.gnu.org/pub/gnu/intlfonts)

- GNU's fonts for English.

Some available free fonts (http://cgm.cs.mcgill.ca/~luc/originalfonts.html)

A large font resource from indic languages. (http://cgm.cs.mcgill.ca/~luc/fonts.html)

Font Pool (http://www.fontpool.com/)

- Various kinds of fonts for english.

7.5. Font creation/conversion/rendering tools

- GNU Font Editor (http://www.gnu.org/software/gfe/gfe.html)

  - Graphical font editor for creating BDF fonts; work is now in progress to support PCF font files.

- Font Forge (previously PfaEdit) (http://fontforge.sourceforge.net/)

  - An outline font editor that lets you create your own postscript, truetype, opentype, cid-keyed, multi-master, cff, svg and bitmap (bdf) fonts, or edit existing ones. Also lets you convert one format to another. FontForge has support for many Macintosh font formats.

- DoubleType (http://doubletype.sourceforge.net/)
  - DoubleType is a graphical typeface designer that builds TrueType font files.

- XmBDFEditor (http://linux.maruhn.com/sec/xmbdfed.html)
  - A Motif-based BDF font editor

- Gfonted (http://www.levien.com/gfonted)
  - A font editor for Adobe Type 1 format fonts.

- Gribouy (http://www.nongnu.org/gribouy/)
  - A font editor for GNOME desktop environment

- Adobe Font Development Kit for OpenType (http://partners.adobe.com/asn/tech/type/otfdk/index.jsp)

- Links to other Font Editors (http://cgm.cs.mcgill.ca/~luc/editors.html)

- Some Proprietary Editors
  - FontLab (http://www.fontlab.com/)
  - Macromedia Fontographer (http://www.macromedia.com/software/fontographer/)

- Font Converters
  - ttf2bdf (http://crl.nmsu.edu/~mleisher/ttf2bdf.html)
    - TrueType to BDF Converter

  - FontConvert (http://www.baraha.com/html_help/baraha/fontconvert.htm)
    - Allows the user to convert the Kannada/Devanagari text from one font to another

7.6. Fonts and Localisation

Localisation makes sense, only if the computer is able to render the localised text. So Fonts play very important role in this. If you are planning to localise an application to your native language, first thing you have to do is to check whether the script used by the language is covered in Unicode or not. If it is covered in Unicode, develop a font for that(if not already available). It is always better to develop a Unicode font rather than a font for a custom encoding.

Notes

1. Two or more characters are printed together when they share common components.
2. As the name implies, all the characters have same width in Fixed width fonts.
3. Serifs are little hooks on the ends of characters. The well known Serif font is "Times New Roman". On the other hand, SanSerif fonts don't have these hooks and look straighter."Verdana" is a famous Sanserif font. Generally Serif fonts are considered more readable particularly when the font size is small.
4. Kerning refers to the process of moving two adjacent characters relative to each other (closer or apart). This is done in order to improve the readability of text, space utilisation on page and produce best fit between the letters.
5. A hint is a mathematical instruction added to the font. These instructions are used while displaying the characters. Hinting decides what pixels to be turned on while displaying the character so that we will get the best possible bitmap image. Hinting is important at smaller resolutions as less number of pixels are available to display a character.
6. Anti-aliasing is a technique for providing smoother looking characters. Scaling of characters results in sharp, jagged edges. Anti-aliasing is the process of smoothening the jagged edges.

Chapter 8. Creation of fonts

8.1. Introduction

(Note: This chapter contains information relating to the creation of fonts. This chapter will give only minimal necessary information that every font designer must know, hence will not act as an exhaustive material on font creation.)

The aesthetics of font creation is the combination of visual appeal and mathematical science. While the visual appeal deals with the ergonomics, mathematics provides a framework to define the behaviour of the fonts temporally.

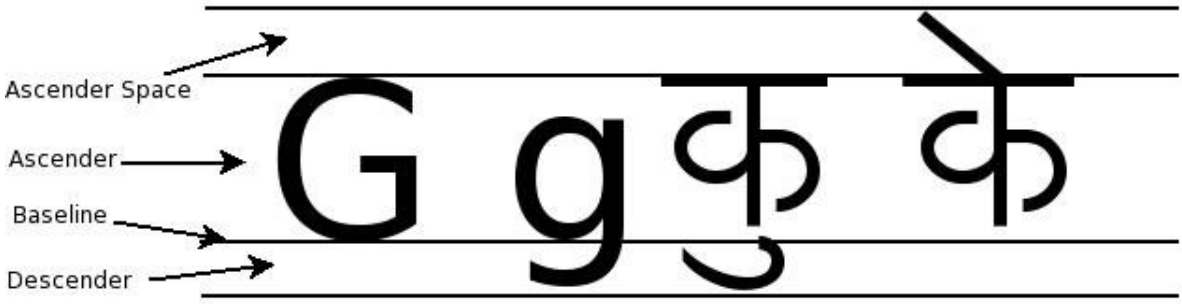
A font is a bitmap image displayed on a computer screen. There are various Font format types like vector format, bitmap format and outline format. Each of these formats store font character data differently, but the final rendering used for printing or screen display is formatted as a bitmap. All the display related information for a particular font set are stored in a font file. These informations are necessary for the fonts to be correctly displayed on the computer screens on different conditions (such as at different screen resolutions). These informations define features like style and arrangement of font character in these conditions.

Almost all console fonts are bitmap fonts, while Postscript, TrueType and OpenType fonts describe the outline of the glyph. The interior of these outline glyphs are filled in by a process known as rasterization. One of the problems with the bitmap fonts is that it is unusable at different scales (i.e, it does not scale well). The scaling problem is addressed well by the outline fonts, in which the theoretical limit is defined only by the environment in which it operates (e.g, limit set by the operating system). Of the three outline font types (PostScript, TrueType and OpenType), OpenType font is the most advanced. Hence our description in this text will emphasise this format more.

Describing a font character necessitates several typographic terminologies. Let us dissect some of them to better understand the theory of creation of the font. Other terms are explained earlier in Chapter 7. Figure below shows some of these notions pictorially.

1. Pixel: Pixel is an abbreviation for picture element. It represents the smallest element in a visual display.
2. Character Cell: Character cell is a square area which is assigned to a character on a computer screen. It is used to depict characters. The dimensions of the Character Cell are decided before the characters are rendered.
3. Baseline: Baseline is an imaginary line on which majority of characters rest.
4. Ascender: Ascender denotes that portion of a character that extend above the baseline (e.g. Uppercase English characters).

5. Descender: Descender denotes that portion of a character that extends below the baseline (e.g. portions of English letters like 'g', 'j', 'p', 'q', 'y').
6. Ascender Space: Ascender space denotes that portion of character which appears above the base character.

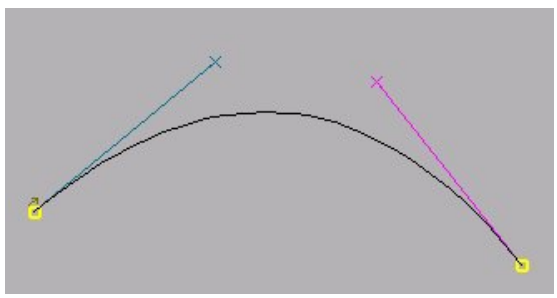


As mentioned earlier, fonts are represented in various formats (outline,stroked,bitmap etc.). TrueType, OpenType and PostScript fonts are described by the outline format.

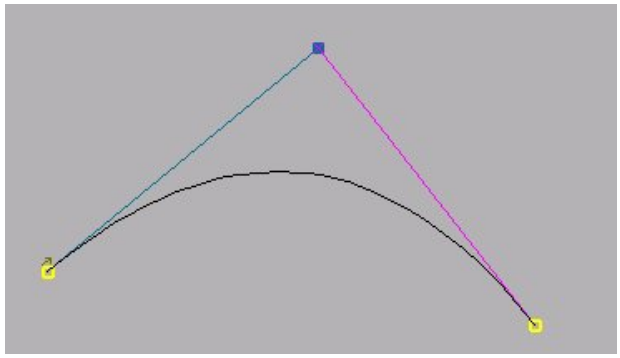
Outline font format is described by a series of mathematical equations, which represent a series of curved splines and line segments. When talking of font technology only two curves are significant. These are quadratic curves and cubic curves. Parametric cubic curves are used to approximate two curves and the line segments.

Each of these splines/curves are described by four points, two of which represent the endpoints of the curve, while the other two, called control points, describe the slope and shape of the curve. These control points are independent in cubic splines, as shown in the figure below.

Figure 8-1.



In quadratic splines, these control points intersect and merge with each other as shown in the figure below.



These control points help in creating the fonts by enabling the font designers to control the curves by moving these control points.

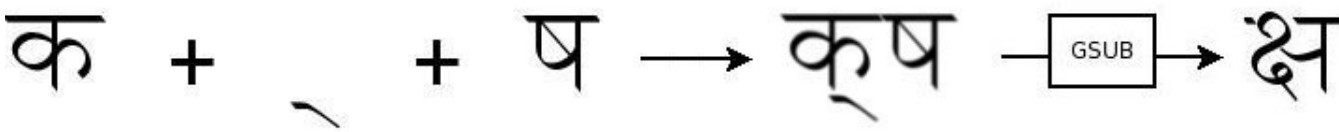
8.2. OpenType fonts

(Note: In this text we will concentrate mainly on OpenType fonts, hence this section describes the major table format supported by OpenType fonts.)

Data in OpenType fonts are contained in table format. These table data comprise either a TrueType or a PostScript outline font data. OpenType font supports most of the advanced features of TrueType and PostScript fonts. One can say that OpenType font is an extension to TrueType which also supports Postscript font data. So if our document uses OpenType font we are actually using either TrueType or PostScript outline in our document. This also means that anything that supports OpenType does not need to convert between PostScript and TrueType.

The tables, contained in the OpenType fonts allow the font creators to embed formatting information in the font itself. These informations are used by the rendering engine to render the font properly on the screen to improve the text layout. At an abstract level, OpenType layout defines five "Advanced Typographic Tables". These tables along with their descriptions are given below.

1. The Glyph Substitution Table (GSUB): The GSUB table stores "substitution information" which can be used to substitute a glyph/glyphs (or ligature/ligatures) with different glyph/glyphs (or ligature/ligatures). For example, consider the character combinations below:



In this example, the first three glyphs combine, to form a group of glyphs. Now the substitution data contained in the font file in GSUB table is applied to these glyphs to form resultant glyph.

The GSUB table supports six types of glyph substitutions.

a. Single Substitution: a single glyph is replaced with another single glyph.

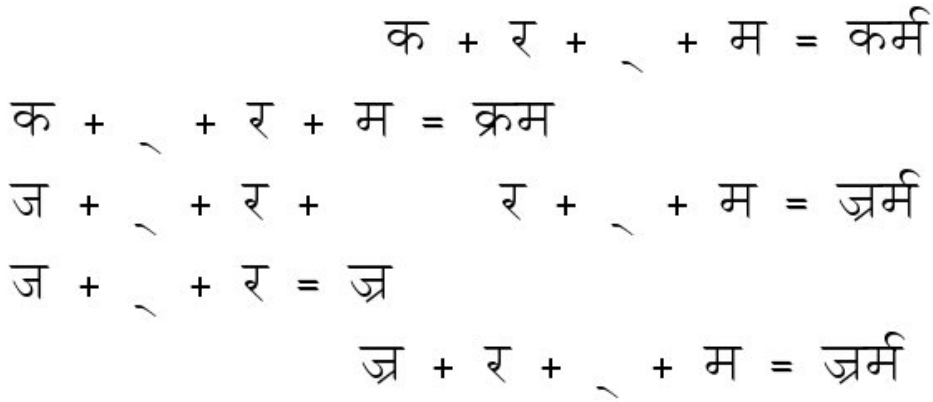
b. Multiple Substitution: a single glyph is replaced with multiple glyphs.

c. Alternate Substitution: a single glyph is replaced with an equivalent alternate glyph.

d. Ligature Substitution: several glyphs are replaced with a single glyph (ligature).

e. Contextual Substitution: glyphs are substituted based on a certain contextual glyph pattern.

f. Chained Contextual Substitution: This type of substitution extends the capability of contextual substitution. Unlike the contextual substitution, the substitution can be applied to the contextual glyph pattern in a chained manner.



The above figure is an example of Contextual Substitution and Chained Contextual Substitution.

2. The Glyph Positioning Table (GPOS): The GPOS table stores "positioning information" which can be used to position the glyphs accordingly or relative to each other based on the GPOS data. For example:

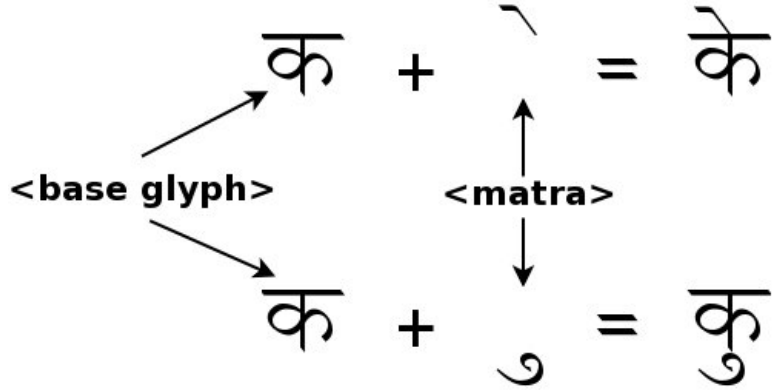
The GPOS table supports eight types of glyph positioning.

a. Single Adjustment: a single glyph is positioned to a location in character cell.

b. Pair Adjustment: two glyphs are positioned with respect to each other. For example all kerned glyphs are pair adjusted.

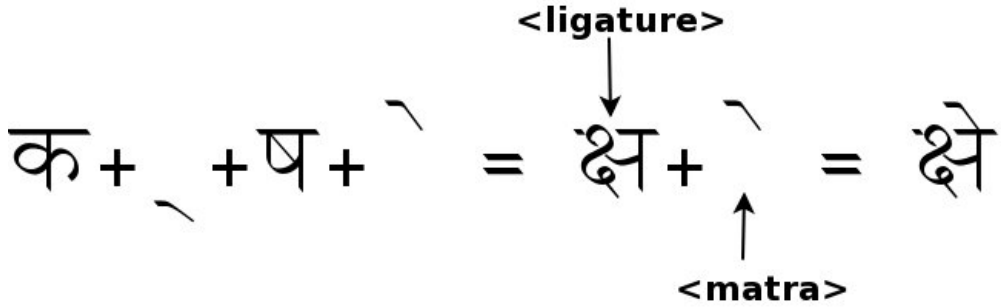
c. Cursive Attachment: the glyphs of those languages which have cursive writing styles are attached together through a common point.

d. MarkToBase Attachment: the combining marks are positioned with respect to the base glyph. For example



In this example the <matra>'s are the combining marks, which combine with the <base glyph>.

e. MarkToLigature Attachment: the combining marks are positioned with respect to the ligature glyph. For example

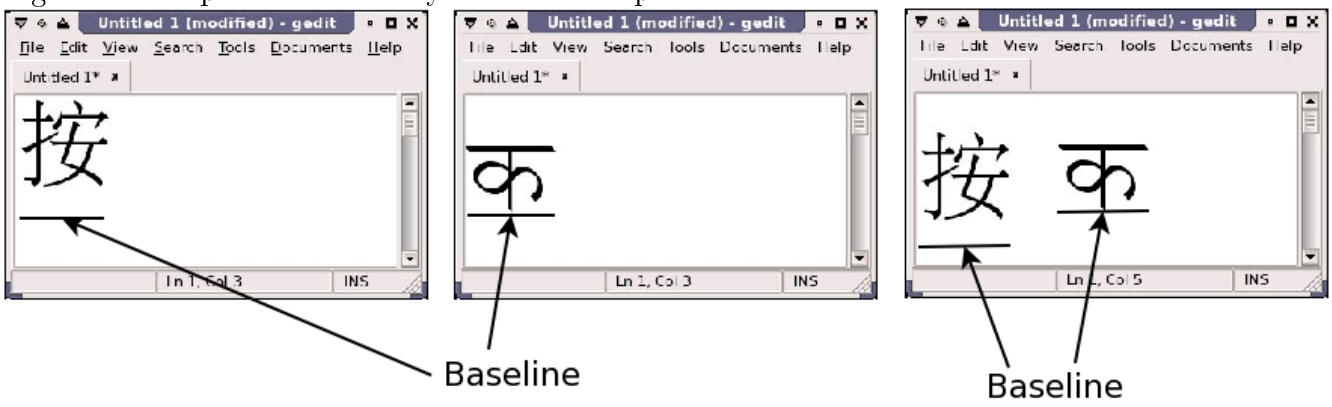


f. MarkToMark Attachment: two combining marks are positioned with respect to each other.

g. Contextual Position: In this type of positioning the glyphs are positioned based on a certain contextual glyph pattern. For example if you specify that if the pattern is of the form <[0-9]+th> raise the letters <t> and <h> as superscripts, then <5th> will be depicted as <5^{th}> after the contextual position is applied.

h. Chaining Contextual Position: This type of positioning extends the capability of contextual position. Unlike the contextual position, the positioning can be applied to the contextual glyph pattern in a chained manner. For example a pattern of the form <$e^{e\ ia}_{z}$>

3. The Baseline Table (BASE): The BASE table provides information which are used to align texts in lines, written in different scripts (texts of different span and sizes), together to improve the text layout. For example

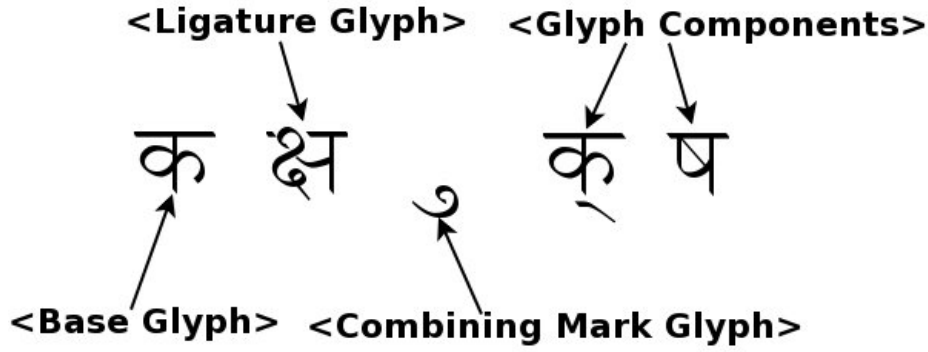


In the above figure, the Chinese character is positioned on a low baseline while the Devanagari character is on a very high baseline. Using the BASE table information, the figure in the final window is aligned properly, to fit on the text layout.

4. Justification Table (JSTF): The JSTF table stores those informations, which can be used by the fonts for glyph positioning and glyph substitution in a justified text.

5. Glyph Definition Table (GDEF): These tables provide informations used by GSUB and GPOS tables to differentiate the types of glyphs in a string. These tables are optional and may or may not be implemented by the font. The GDEF table contains three types of information in in subtables. These are-

   a. GlyphClassDef: The Glyph class definition table stores information related to different glyph classes.



These are generally of following types.

i. Base Glyphs

ii. Ligature Glyphs

iii. Combining Mark Glyphs

iv. Glyph Components

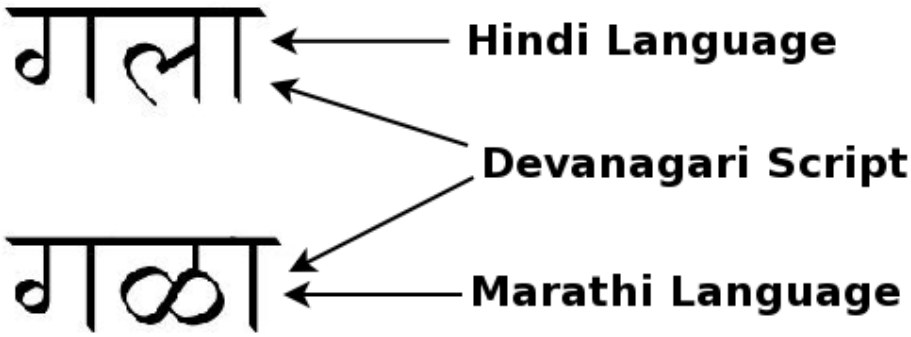
b. AttachmentList: The Attachment Point List table stores information related to the attachment points of a glyph. These informations can be cached and hence need not be recalculated each time the glyph is displayed.

c. LigatureCaretList: The ligature caret list table defines the caret positions in the ligature. These caret positions are fine tuned for different scales.

The typographic data for GSUB and GPOS tables are organised in a similar manner. These typographic informations are organised as Script, Language System, Feature and Lookup tables.

1. Script: A Script table stores information for a collection of glyphs which are used to represent one or more languages (e.g. Devanagari is used to write Hindi and Marathi). Hence a Script represents a set of Language Systems.

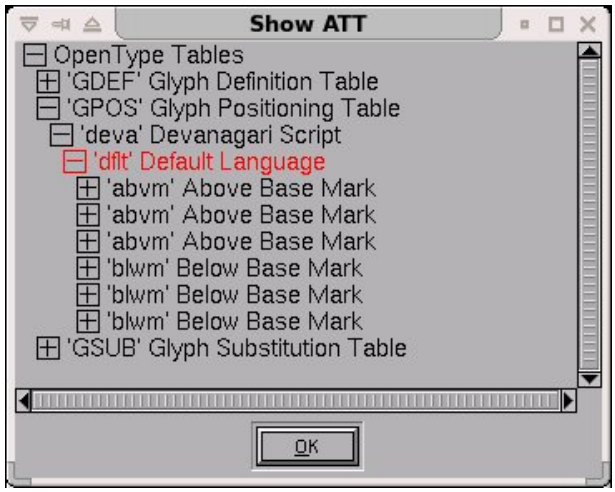
2. Language System: A Language System table stores those informations which are used to modify the function or appearance of glyphs for a particular language. For Example



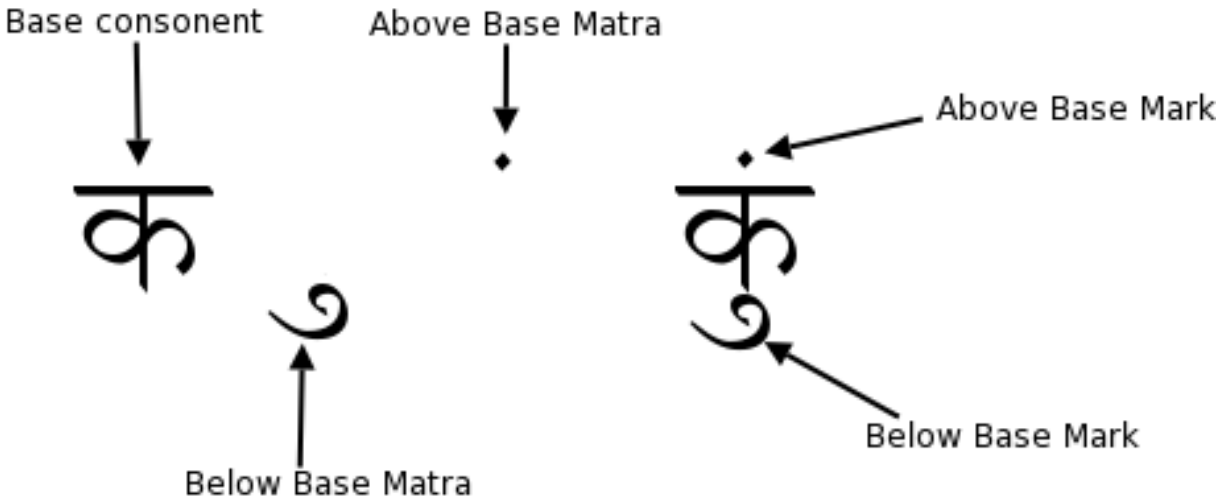
In the above figure both words (Hindi and Marathi) are spelt the same (also semantically same but pronounced differently). But in Marathi they are written with a special glyph, as shown in the figure below.

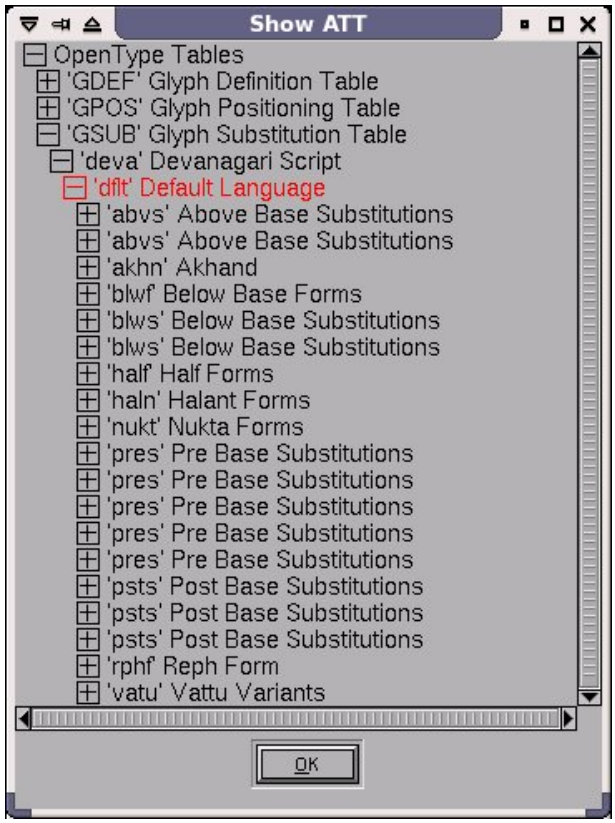
A Language system contains a list of Features.

3. Feature: A Feature table stores information which is used as typographic rule for a glyph. A collection of such rules are applied on glyphs to represent a language. In the absence of specific rules, the default rules are applied. [OTF2] contains a list of registered Features. The figures below show an example of such Features implemented in a Devanagari Font [Raghindi, Copyright NCST].



The above figure shows the default language Features implemented for GPOS table, for Hindi language (Devanagari Script).

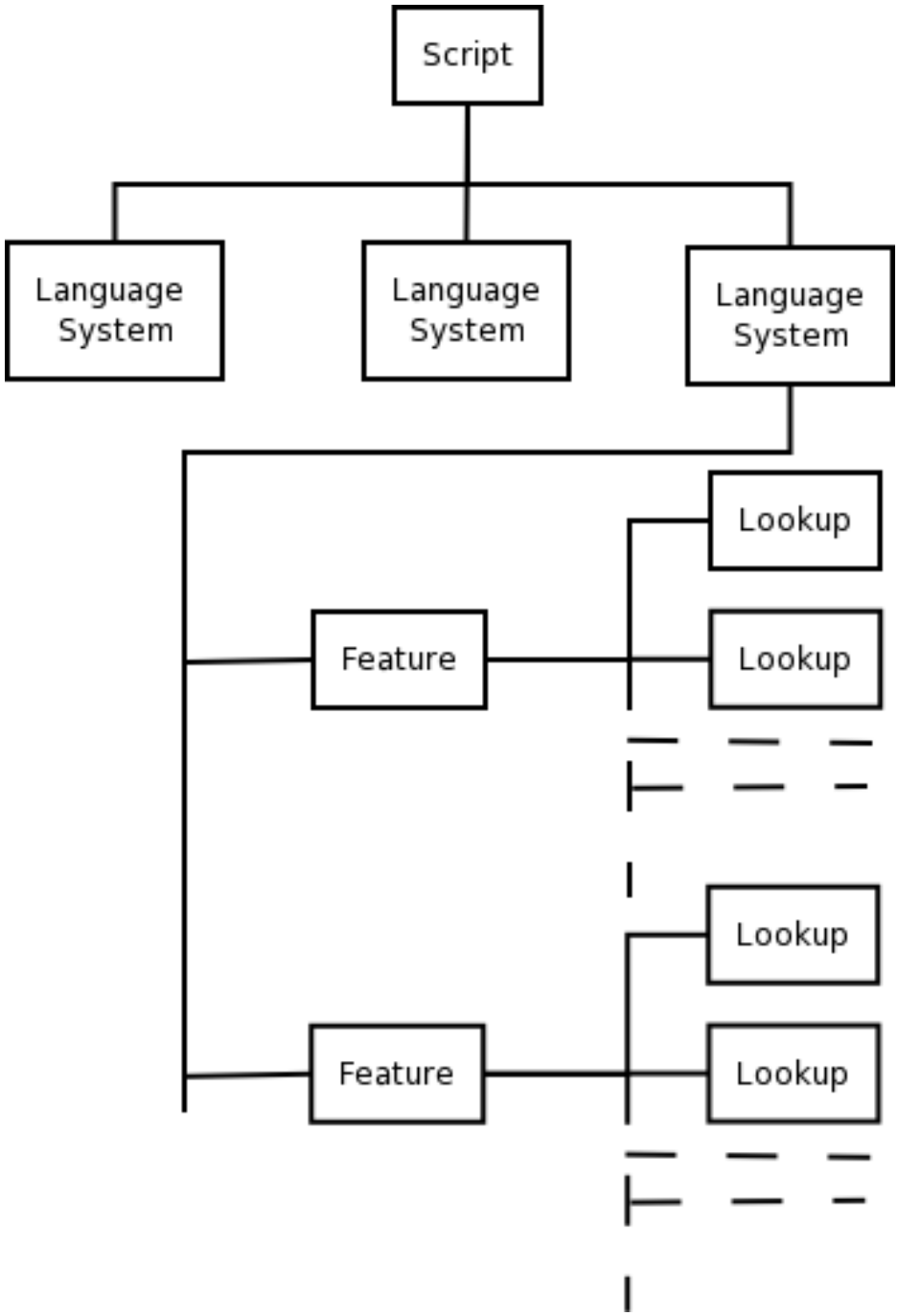




The above figure shows the default language Features implemented for GSUB table, for Hindi language (Devanagari Script).

4. Lookup: A list of Lookup tables implement a Feature by applying the information contained in them. These tables contain only one type of information. This information depends on whether the Lookup is a part of GSUB or GPOS table. GSUB supports six Lookup types where as GPOS supports eight Lookup types (described earlier).

At an abstract level these typographic information are described in the following figure.



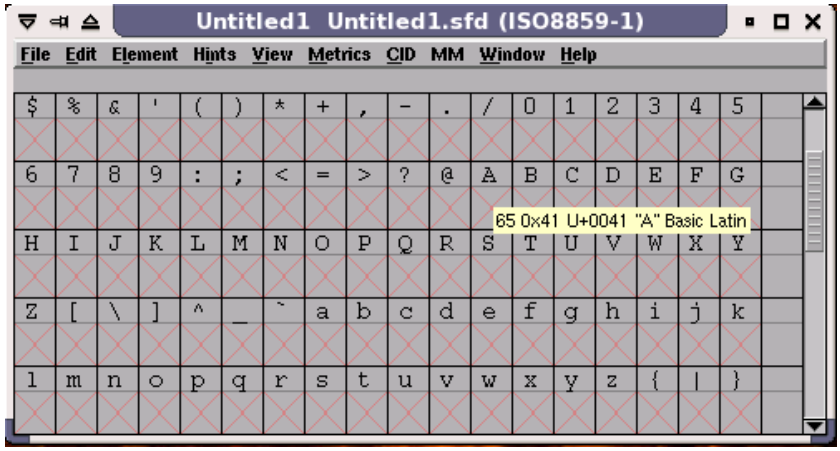
This figure is not precise and just gives an overview of the arrangement of OpenType Font tables. The full explanation of OpenType specification is beyond the scope of this text. The interested users can look at [OTF3] to read the full specification.

8.3. How to create fonts

In order to create fonts, the first thing that we require is a generic font editor which is able to support editing in multiple font formats and multiple character encodings. In this text, we will use fontforge font editor to demonstrate the creation of fonts. We would not go into much detail, and cover only the major points, as font creation is

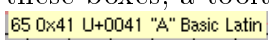
covered in detail in the Fontforge manual which is available online (http://fontforge.sourceforge.net/overview.html) [FOT18]. To create fonts there are also other general purpose font editors which are listed in the tools section of this guide.

The figure below shows the default view of fontforge editor when it is opened.

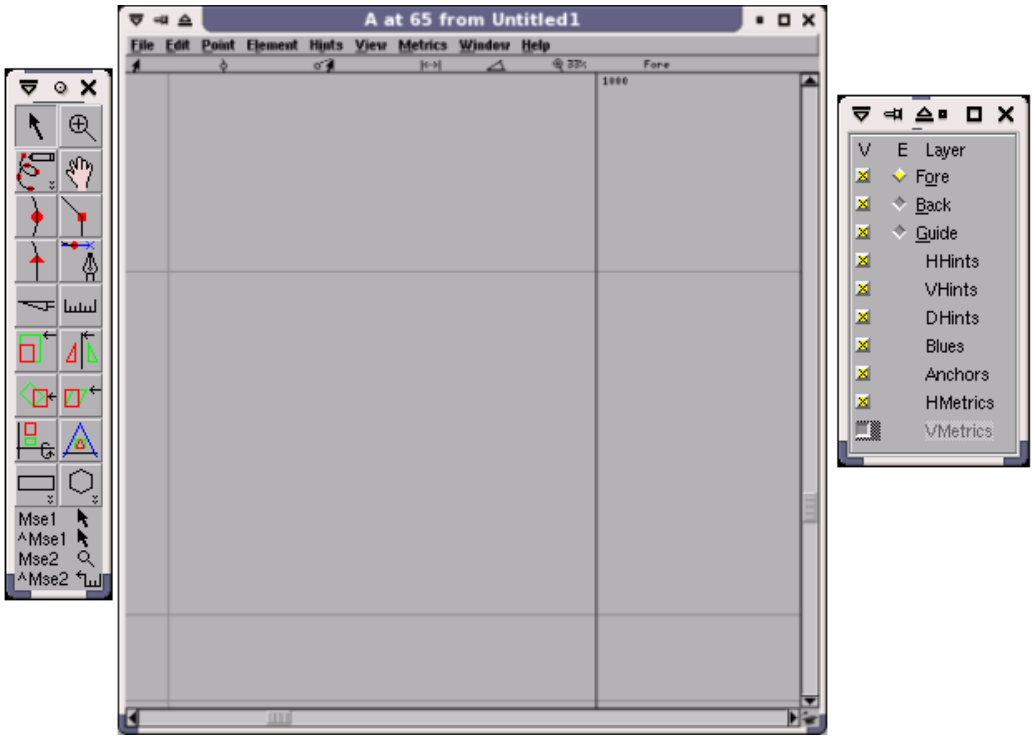


The above figure gives a birds-eye view (also known as Font View) of the Fontforge application. In this view there are small boxes as shown below,

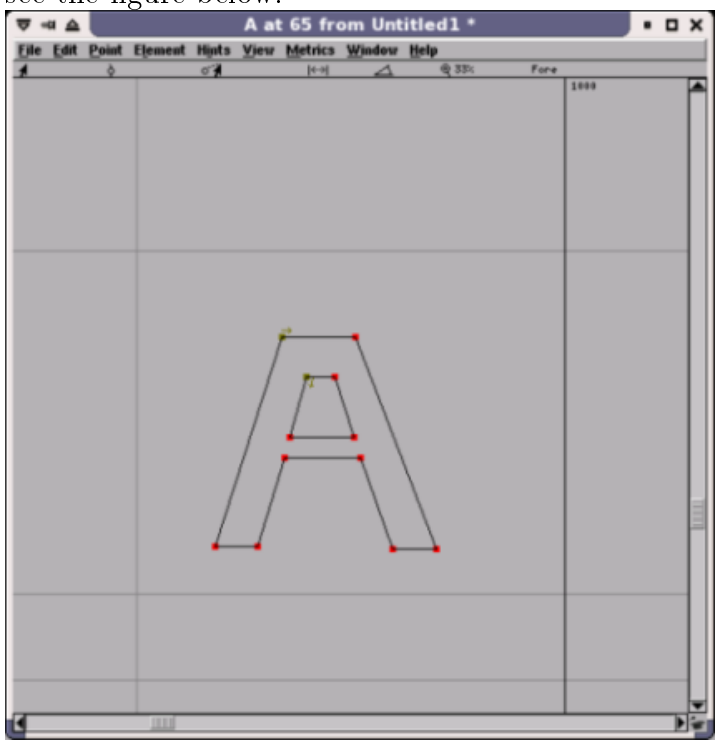
which are divided into two halves. By default the lower half is crossed (in red colour), while the upper half is marked with character label (which denote the character that glyph represents, in lower half), N in this case. When we put our mouse over any of these boxes, a tooltip pops up as shown below.



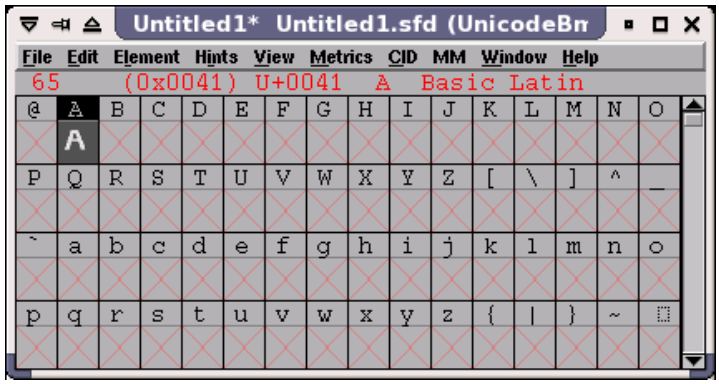
This tooltip gives us informations like glyph number, Unicode number, etc. Now when we double click on any of these boxes, it opens a window, as shown in the figure below:



This window, also known as outline glyph view, is where we create the outline of a glyph for a particular Unicode Codepoint/ligature character. The Font View now, displays these outline glyphs in place of red crosses. We can use the tools provided with the leftmost window, as shown in the above figure, to create the glyphs. For example see the figure below.



The glyph created here is for the Unicode Codepoint 0041, which is character A of English language.



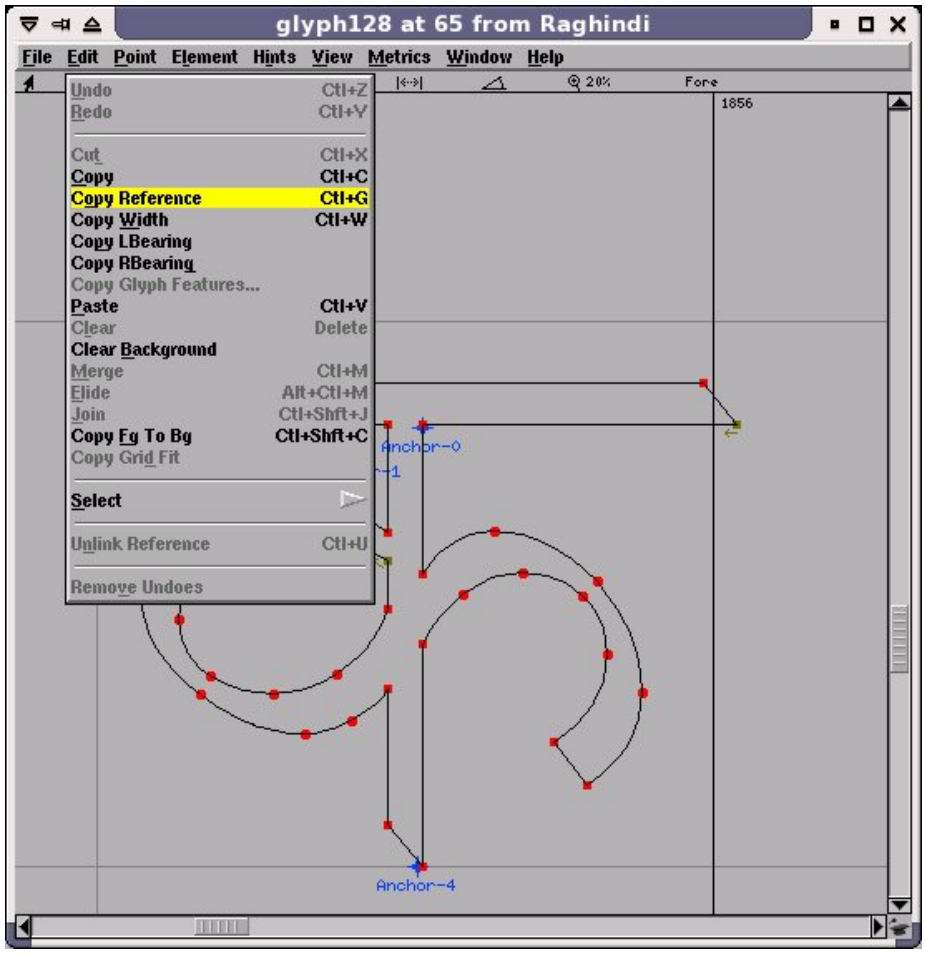
The Font View now looks like the above figure.

Now we will discuss the following topics:

1. Creating References
2. Creating Anchorpoint
3. Ligature
4. Kerning

8.3.1. Creating Reference

Open any glyph and click Edit|Copy Reference (as shown in the figure below).



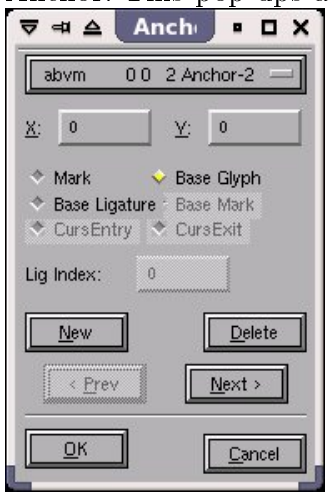
Open any other glyph window and click Edit|Paste. This will create a Reference to the above glyph into another glyph for another Codepoint/glyph. This saves a lot of space as the characters can now be made up of simple glyphs which are stored somewhere else- once per glyph, irrespective of how many characters use the glyph.

8.3.2. Creating AnchorPoint

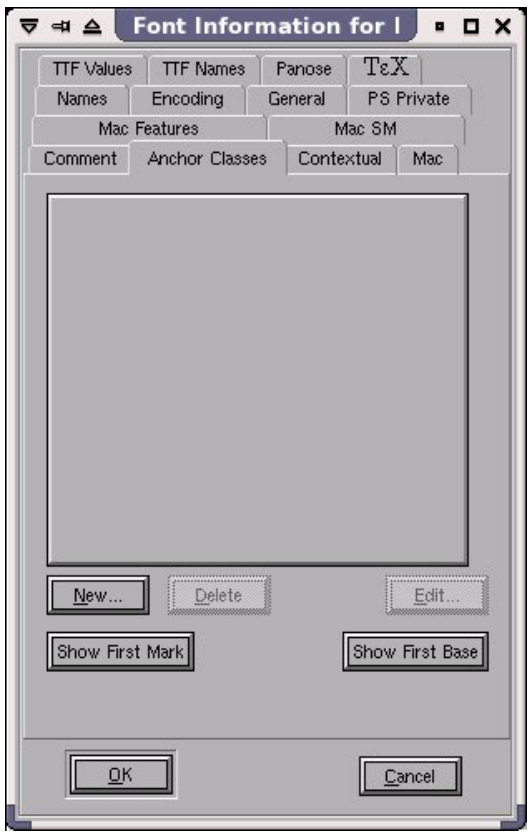
Anchor Points are used to join two glyphs in a font.

For example, both the glyphs in the above two figures define anchor point Anchor-1. When these two glyphs are combined, they join at the anchor point Anchor-1 to produce:

To add existing anchor points to any glyph, click (inside Glyph Menu) open Point|Add Anchor. This pop-ups a window, as shown below



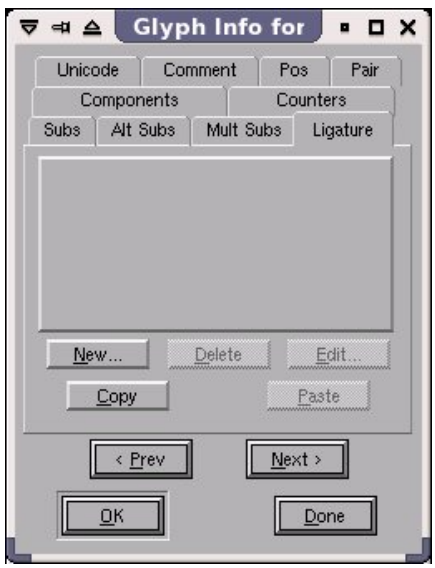
Then to create an anchor point, click Element|Font Info. This will open the Font Information window as shown in the figure below.



Then click Anchor Classes|New, which will bring the window as shown in the figure below.

8.3.3. Ligature

Ligatures are a combination of two or more glyphs. The ligature may have a shape which is a combination of the three shapes or a partially/totally different glyph. A glyph may act as a Ligature glyph. To add Ligature information to this glyph click (in glyph Window) Element|Glyph Info|Ligature. This will pop-up a window, as shown below

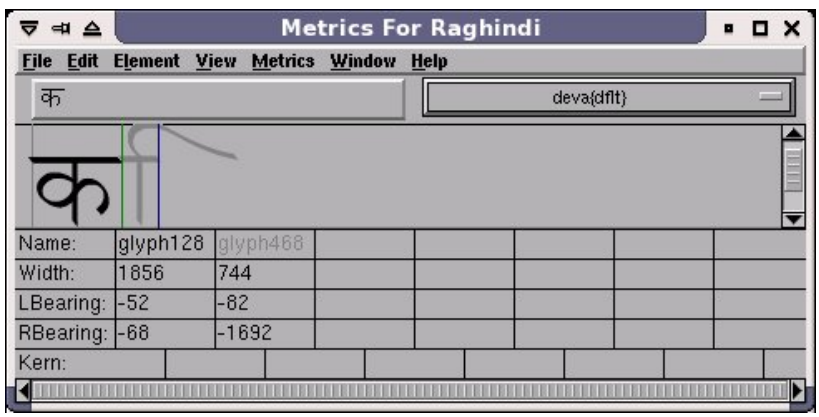


Now in the above window click New, which will pop-up a window as shown in the figure below.

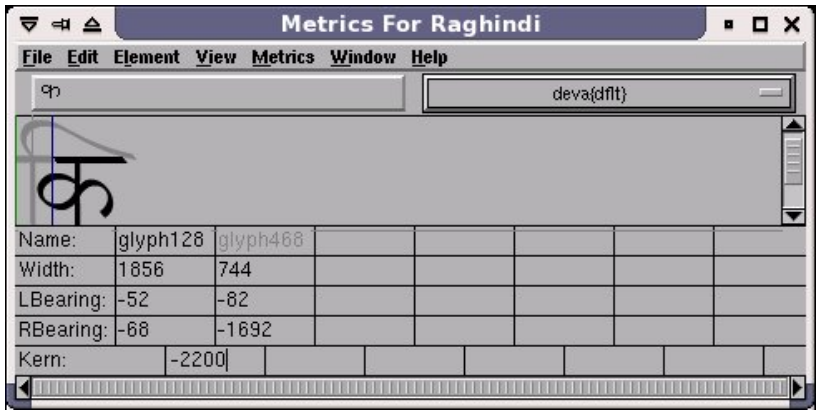
In the textbox labelled components, add glyph names which are part of the ligature, separated by a space. The glyph for which this information is added is now a ligature for these glyphs.

8.3.4. Kerning

Kerning refers to the process of moving two adjacent characters relative to each other (closer or apart) for better readability, etc. In order to add kerning information, open Metrics window by clicking (on the main Window) Window|New Metrics Window. This will pop-up a window as shown below.



Fill the appropriate information in the text labelled kern. The final widow looks like as shown in the below figure.



8.3.5. Conclusion

Font creation is a vast topic in itself (we also need to have an understanding of the thousand years of language development), and we have just touched the surface. Interested readers can go ahead and explore further. The URL's provided in this chapter may be useful.

Chapter 9. Input Methods

This chapter describes the input methods in Linux family of softwares. The description is aimed at a naive user who wants some information on Input Methods. We will explain the principles and wherever necessary appropriate examples will be included. If the user feels interested he can go ahead and explore further. The Reference section will help such user to explore further.

This document addresses two types of users:-

- A user who simply wants to enter text in his native language.
- Developers who want to use existing libraries to implement/localise their softwares in a particular language say hypolan.

9.1. Introduction

Read if: If you want to get an overview of various Input Methods available on Linux platform.

We need the ability to input texts from keyboard when using computers. This ability is already there for international language like English. But when we speak in terms of localisation, we must enable these Input Methods for our respective languages. They are not available by default and we must either re-invent them from scratch or use the existing ones.

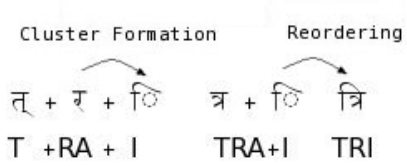
Input Methods are concerned with all those aspects which deal with entering text on the computer screen. These range from layout description of your keyboard to writing texts in your native language. The text is input through the keyboard which is mapped to a character in a language using some conversion-dictionary/keymap.

But why do we require Input Methods at all? Why don't we require it for languages like English. All characters for language such as English (mainly Latin) can be input using a single keyboard. Keymap is all that a user has to care. (Keymap defines how the keyboard input/output is processed for a particular language.)

There are certain languages which have thousands of characters, and a direct keymap in combination of other keys(probably dead keys) is not possible. (Dead keys are those keys which, when pressed, produce no output and work only in association with other keys. These keys change the output of the key that is pressed immediately after them.). And there are languages which require preprocessing before they can be displayed on the computer screen. For language like Hindi(using Devanagari script) when the letters

are put together they combine(form cluster) together and either reorder/transform themselves to produce a shape which is different from the original series of characters.

For example, the figure below shows a sequence of three keystrokes and the resulting combined character.



Then there are ideographic languages (common example CJK family=Chinese Japanese Korean) where each character represents actual objects and not sounds, resulting in a large character set. There are also provisions for phonetic literals in these languages which are entered as inputs on the computer screen. Since several ideographic characters in these languages have the same phonetic sound, users must be given the option to choose from within these possible outcomes. The Arabic scripts are written in Right-to-Left direction, which violates even the natural cursor movements.

We also want our editors to handle texts in multiple languages at the same time. We need methods to switch between different languages, while editing a multilingual text in any such editor,either by using key combinations or any GUI method. We also want cut-and-paste facility across the editors.

All these scenarios highlight the need for powerful and generic interface to handle input/display of characters in various languages. Input method takes a step to solve these problems.

We will discuss the following framework in the rest of this chapter.

1. X KeyBoard Extension (XKB)
2. X Input Method (XIM)
3. GTK+ IM
4. IIIMF Framework

9.2. Who uses what?

The Linux graphical environment is a client-server system. The clients (e.g. X clients) and servers (e.g. X servers) communicate by means of events based on some protocol (X protocol). A library (X library or Xlib) encapsulates X protocol and provides a set of

API's, so that clients can do everything in terms of function calls. Internally every framework uses X protocol and Xlib, as these are now standards. There are some middle layer libraries called toolkits(e.g. GTK+ and Qt) which ease the use of low-level Xlib primitives.

XFree86: XFree86 uses X Input Method(XIM) in order to input texts and X keyboard(XKB) Extension to describe the keyboard map.

GTK+: GTK+ uses it's own input method framework called GTK+ IM. In this arrangement input method modules can be dynamically plugged in as per user commands.

Qt+: Qt uses X Input Method (XIM) in order to input texts.

Apart from these there is there is an IIIM Framework which is a Windowing System independent, multilingual and multiuser input method framework.

9.3. X Keyboard Extension

[Note:- Please locate your xkb directory. Normally it is /etc/X11/xkb. Let us call it <XKBdir>. All description in this section will be relative to this directory.]

One of the problems with X protocol core library is its limitations when it comes to extending the keyboard behaviour with respect to the changing conditions. This is because of the strict specifications it has been put through and the keyboard cannot be imparted additional behaviour with the changing conditions. The XKB tries to solve these problems as it can be used to specify keyboard's behaviour on a per key basis. Let us first define some concepts then we will look at an example.

keycodes

These are the tables which define the symbolic names for key scan-codes (http://www.barcodeman.com/altek/mule/scandoc.php) [IMS7]. key scan-codes are the raw keyboard signals that are passed from the Linux virtual terminal to the X Server without any translation. These keycodes are located in <XKBdir>/keycodes directory (refer to <XKBdir>/keycodes/README).

e.g.

```
<TLDE>= 49;
<AE01>= 10;
```

These are the interpretations of the raw key scan code reported by the keyboard device.The key scan code of the key is translated into keycode according to the keyboard model. These may be according to the specification of different manufacturers (and hence may contain entries like IBM,SONY,Sun, but usually the map is xfree86). These keycodes are further used to map Unicode characters with keyboard keys.

types:

types defines how a key symbol (described below) is produced by a keycode change when modifiers (e.g. shift,control etc) are applied. The key types are located in the <XKBdir>/types. By default there are four predefined types which are described in the file <XKBdir>/types/basic

1. ONE_LEVEL :- When this type is applied on any key, there is only one key symbol value (one level) for that key, and modifier do not have any effect. For example,usually if we press Enter, Escape, Space, etc we will have the same character irrespective of any modifier key.
2. TWO_LEVEL :- When this type is applied on any key there can be two key symbol values (two levels) for that key. The second level is selected by pressing Shift modifier, but these types don't depend on the LOCK modifier state. (e.g. [1,!], [2,@] ,[3,#] etc., second character in each pair is achieved by pressing Shift key).
3. ALPHABETIC :- When this type is applied on any key there can be two key symbol values (two levels) for that key. The difference from the earlier TWO_LEVEL class lies in the fact that these keys depend on the lock modifier's state too.
4. KEYPAD :- When this type is applied on the keys on the keypad there can be two key symbol values (two levels) for that key. These keys depend on the two modifiers state namely NumLock ans Shift. The difference from the earlier ALPHABETIC lies in the different modifiers applied.

The file <XKBdir>/types/basic is listed below.

```
// $Xorg: basic,v 1.3 2000/08/17 19:54:48 cpqbld Exp $
default xkb_types "basic" {

    // Fairly standard definitions for
    // the four required key types

    virtual_modifiers NumLock;

    type "ONE_LEVEL" {
modifiers = None;
map[None] = Level1;
level_name[Level1]= "Any";
    };

    type "TWO_LEVEL" {
modifiers = Shift;
map[Shift] = Level2;
        level_name[Level1] = "Base";
      level_name[Level2] = "Shift";
```

```
    };

    type "ALPHABETIC" {
modifiers = Shift+Lock;
        map[Shift] = Level2;
        preserve[Lock]= Lock;
        level_name[Level1] = "Base";
        level_name[Level2] = "Caps";
    };

    type "KEYPAD" {
     modifiers = Shift+NumLock;
     map[None] = Level1;
     map[Shift] = Level2;
     map[NumLock] = Level2;
     map[Shift+NumLock] = Level1;
     level_name[Level1] = "Base";
     level_name[Level2] = "Number";
    };
};
```

compat:

These files describe how keyboard state changes when you press one of the modifier keys. XKB has some internal variables which define how symbols are produced when a modifier is pressed.

Symbols:

These tables map each symbolic keycodes into the character encoding for a particular language. For every keycode all possible symbols are specified which, depend on the key types (described earlier). Which means that for each specific case of keycode the symbol values depend on the modifier state and their behaviour. The behaviour of these modifiers are defined in the files in the directory <XKBdir>/compat/. symbols are located in <XKBdir>/symbols directory. Each supported language finds its entry in this directory. If you don't find an entry for your language, then you will have to create one(described later).

Creating a keyboard map for your language:

The version of Linux that you are using might not have support for your language. That is to say that the keymap for your language is currently not available. Hence we need to create one file for this language in the <XKBdir>/symbols directory.The Xfree86 4.3.x default rules (rules files are located in <XKBdir>/rules)uses files inside <XKBdir>/symbols/pc. The difference between the old files in <XKBdir>/symbols and new files under <XKBdir>/symbols/pc is that under the new files directory,only one group per file is defined. A group is a logical state of a keyboard providing access to a collection of graphic characters. These characters are arranged in several levels (see types above). For example

There can be multiple keyboard groups associated with a keyboard. Each symbol set has its own group. When we change the group, it means we are selecting a different set of symbols. Each of these groups have different shift levels. This property can be utilised to switch between the different languages (e.g by using setxkbmap, as described below.), in order to input text in these different languages.

Table 9-1. Keyboard group and shift level as a rectangular matrix

	Level1	Level2
Group1	a	A
Group2	ae	AE

For Devanagari scripts the keymap file is named as dev in the <XKBdir>/symbols directory, if we do not find one then we will have to create one. A sample format for such file is described below.

Table 9-2. An example of file <XKBdir>/symbols/pc/dev

```
// based on a keyboard map from an 'xkb/symbols/dev' file
//
// $XFree86: xc/programs/xkbcomp/symbols/pc/dev,v 1.2 2002/11/22 04:03:28 dawes E

partial default alphanumeric_keys
xkb_symbols "basic" {
      name[Group1]= "Hindi";

      key <TLDE> { [      0x100094A, 0x1000912          ]       };

      // Mainly numbers.
      key <AE01> { [      0x1000967, 0x100090D          ]       };
      key <AE02> { [      0x1000968, 0x1000945          ]       };
      key <AE03> { [      0x1000969                     ]       };
      key <AE04> { [      0x100096A                     ]       };
      key <AE05> { [      0x100096B                     ]       };
      key <AE06> { [      0x100096C                     ]       };
      key <AE07> { [      0x100096D                     ]       };
      key <AE08> { [      0x100096e                     ]       };
      key <AE09> { [      0x100096F, parenleft          ]       };
      key <AE10> { [      0x1000966, parenright         ]       };
      key <AE11> { [      0x1000903                     ]       };
      key <AE12> { [      0x1000943, 0x100090B          ]       };

      // Mainly long vowels

      key <AD01> { [      0x100094C,  0x1000914         ]       };
      key <AD02> { [      0x1000948,  0x1000910         ]       };
      key <AD03> { [      0x100093E,  0x1000906         ]       };
      key <AD04> { [      0x1000940,  0x1000908         ]       };
      key <AD05> { [      0x1000942,  0x100090A         ]       };

      // Mainly voiced consonants

      key <AD06> { [      0x100092C,  0x100092D         ]       };
      key <AD07> { [      0x1000939,  0x1000919         ]       };
      key <AD08> { [      0x1000917,  0x1000918         ]       };
      key <AD09> { [      0x1000926,  0x1000927         ]       };
      key <AD10> { [      0x100091C,  0x100091D         ]       };
      key <AD11> { [      0x1000921,  0x1000922         ]       };
      key <AD12> { [      0x100093C,  0x100091E         ]       };

      // Mainly short vowels
      key <AC01> { [      0x100094B,  0x1000913         ]       };
      key <AC02> { [      0x1000947,  0x100090F         ]       };
      key <AC03> { [      0x100094D,  0x1000905         ]       };
      key <AC04> { [      0x100093F,  0x1000907         ]       };
      key <AC05> { [      0x1000941,  0x1000909         ]       };

      // Mainly unvoiced consonants

      key <AC06> { [      0x100092A,  0x100092B         ]       };
      key <AC07> { [      0x1000930,  0x1000931         ]       };
```

In this example each keycode has two values, first one when the Shift modifier is not pressed and the second one when the shift modifier is pressed. These values are the Unicode values prefixed with 0x100. For example the letter <2309> [DevanagariA] (if you cant see this letter,read the letter read A Simple Example)with Unicode value of 0905 is written as 0x1000905. Values like key <AE01> are defined in<XKBdir>/keycode for your particular keyboard layout. Some symbols may be predefined for Xlib in <X11/keysymdef.h> . Hence you can use predefined symbols in place of Prefix+Unicode values.

To support your language all you have to do is to create a file like this, in the <XKBdir>/symbols.dir file. This xkb keyboard description is compiled using xkbcomp. In order to do this change directory to <XKBdir> and simply invoke the following command:

```
xkbcomp -lhlpR "*" -o ../symbols.dir
```

For more details see the manual page for xkbcomp.

A Simple Example

(Note:After reading through this example readers will have an idea as to how to input texts in their native languages using the xkb framework.)

The steps required to enable input method for a particular language are more or less same when using XKB extension. Before moving further these things must be checked :-

1. Check to see if proper fonts are in place for the target language. If not then either the scripts will probably be displayed as ???????? or [][][][][][][][][] or any other garbage value. Fonts installation and configuration is described here(link which describes the installation of font). Fonts creation is described here(links to the creation of fonts).
2. Set the locale for your fonts. The locale related issues are described here(links to the locale related issues).
3. Check to see if there is a keymap available for your language(described above). If not then you have to create one . In simple terms you need to create symbols map/keymap (described above) for your language, like the languages defined in <XKBdir>/symbols.
4. Unless support for older versions of XFree86 is required, all you need to do is prepare a single group file under <XKBdir>/symbols/pc directory. For older versions, the groups are precombined and located in <XKBdir>/symbols itself.

When the above setup is done you can manually install the keymap using the setxkbmap command but this will have to be done each time that you log in to X.

Therefore arguments for setxkbmap can be specified in /etc/X11/XF86Config file so that every time the X Server starts, these setups are done. This can be done by describing it in "InputDevice" section for keyboard in /etc/X11/XF86Config file.

After you are through with all these, simply invoke the command like this:-

setxkbmap <optlan1> <optlan2> .... -option grp:alt_shift_toggle,grp_led

where optlan1, optlan2.....optlank are names of language files described in symbols subdirectory.

For example

```
setxkbmap us,kan,ml,dev -option grp:alt_shift_toggle,grp_led:scroll
```

This sets the English, Kannada, Malayalam and Devanagari as the input methods, so you can switch between these languages and type the text in the desired language. The switching is done in this example by pressing alt+shift keys. The Scroll Lock LED indicator on the keyboard will indicate that the languages are scrolled through. It will glow on for the selection of second language onwards till the first is reached again. For more information, see the manual page for setxkbmap.

9.4. X Input Method(XIM)

X Input Method (XIM) is a generic API for Linux applications which can be used by application developers to create internationalised programs and which can be localised as per the requirements of the native languages. XIM framework depends on the locale.

There are two models that are used to implement X Input Method Client/Server model, and Library model.

XIM is a good option for applications,which deal with complex languages (e.g. CJK family,indic family languages), which require some kind of keystroke interpretations to obtain characters. This is why XIM is implemented under Event Handling approach. Events are reported through messages. Keystrokes are sent as events(key event) by the X Server to the IM Library or IM Server.

Client/Server model

The server (IM Server) acts as a separate process which processes the keystrokes coming as events. All the inputs from the X Server are delivered to IM Library which delivers it to the IM Server. The IM Server handles all kinds of preediting and committing.The X Library which also acts as the client to IM Server receives all committed strings.There are two methods to deliver inputs to the IM Server.

1. BackEnd Method: In this method, the client window input events are delivered to IM Library which directs it to IM Server. There is no synchronisation problem because all events are handled serially in the order they are delivered. This method is the default method of implementation.

2. FrontEnd Method:- This method is created to boost the performance. In this method the events are delivered by the X Server to both IM Server and IM Library simultaneously. This also leads to the synchronisation problems such as loss or duplication of key events. This is why BackEnd Method is the core method supported and FrontEnd method is there as an extension to boost the performance when required in specific cases.

All those languages which require complex preediting of the inputs are implemented using Client/Server model.

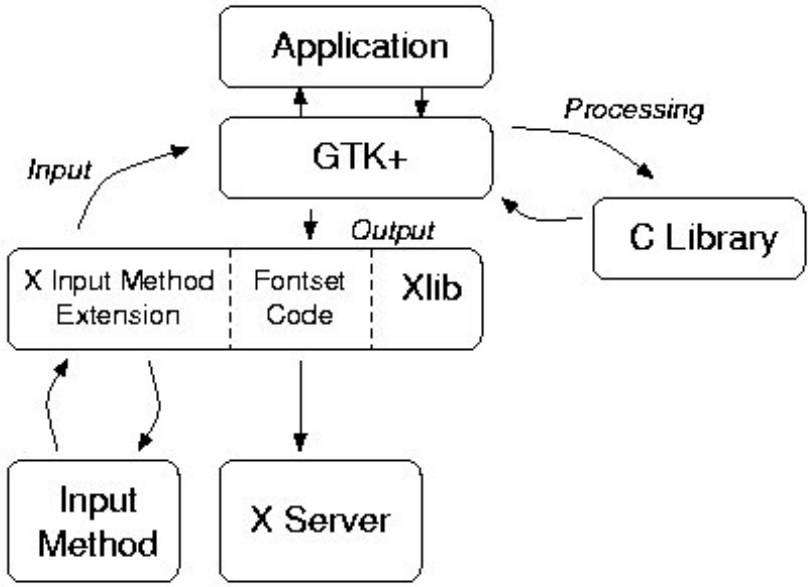
Library Model

All Inputs sent by X Server is handled by the IM Library within the application. The separate IM Server is not required as complex preediting is not supported. All those languages which don't require any complex preediting are implemented using this model (e.g. Indic family Languages and CJK family languages).

9.5. GTK+IM

In GTK+ multilingual support is achieved through modularisation. This means that the Language modules are loaded on demand. GTK+ defines its own framework as a cross platform toolkit. This is done so that GTK+ does not rely on any specific platform (like UNIX/Linux). There are certain drawbacks to this approach. The Input Method is specific to the GTK+ and hence cannot be shared with non-GTK+ applications. However one can use the more recent IIIMF framework (described later) to support Input Methods across platforms.

Figure 9-1. Input Method Framework used by GTK+ application.



The diagram above illustrates the Input Method framework used by GTK+ applications. Text is entered in any GTK+ application using X Input Method Extension (XIM). The key press by the client is sent as an event to the Input Method. The Input Method can call the client by sending some signal. The client provides some handlers for these signals in terms of some functions. These handlers provide all the features for entering text in any complex language. For more information please refer [IMW2] and [IMW3].

9.6. IIIM Framework

(Note: While explaining iiim framework, we have tried not to put too much technicalities to the users. Those who need more details are adviced that after reading this text, they visit the appropriate links specified in the text. The impatient users can start straight here (http://www.openi18n.org/) [IMH1] and start at the IM(IIIMF) subsection (branched under a treelike structure) listed under the heading Sub-committees at the top-left corner of the page.)

9.6.1. Why Another Input Method?

One may ask, why do we need another Input Method, when we have a sophisticated Input Method like XIM. Several reasons can be attributed to the mistakes done in XIM design. These mistakes were publicly accepted by Hideki Hiura, who designed X Input

Method (XIM) as well as IIIM Framework.We briefly discuss some of these mistakes/problems in the following paragraphs.

Platforms other than X could not utilise XIM as it is designed only for X Window System. This is strictly against the philosophy of distributed computing on heterogeneous platforms. Because the current trend shows that distributed computing will be the future of computing, an input Method which cannot federate existing Input Methods, in a coherent and platform independent way is undesirable.

In X Input Method, for each user client Application a new XIM server process is forked. For instance, for five client application five different XIM servers are required. Because XIM is designed on per user server basis, it puts a lot of burden on systems low in resources. XIM also requires a very high network bandwidth which is again against the philosophy of distributed computing. To utilise the real power of distributed computing, our applications should be light weight, evenly fragmented across distributed environment federating it's different responsibility to different components of the application. This will decrease the overall traffic between the client and server, which in turn will decrease the overall traffic on the network.

XIM does not provide any generic Input Method Server Framework, which has resulted in the development of many monolithic implementations of XIM Servers (for particular languages). Due to this, there are different implementations for each language, for which the server is implemented, and if there is a need to support any other language, than present, there is no framework to do that. That may lead to modification and recompilation of the source, or in the worst case development from the scratch.

9.6.2. IIIMF Architectural overview

The evolution of Input Method has been platform dependent and operating system dependent. Each platform has its own way to enter text in localised languages. There is a need for a platform independent, distribution independent, distributed way for entering text using some Input Method.

IIIM Framework is a direction towards achieving platform neutrality when it comes to entering texts on different computer platforms. It is a collection of software components that enable a user to enter text in ways other than simple typing on keyboard on a distributed platform. It also solves the input Method requirements of complex languages like Chinese, Japanese and Korean(CJK).

The platform neutrality is achieved by providing a single distributed wrapper of IIIM Framework over different native Input Method frameworks. This framework integrates a variety of different Input Method Engines which implement input method on these platforms . These engines facilitate the conversion of Input Events from the keyboard to language character sets on these platforms.

There are multiple IIIM Servers (under IIIM Server Framework), each acting as agent, which appear as a single virtual server to IIIM Clients (under IIIM Client Framework).

The Client and Server communicate through IIIM Protocol, which itself is platform neutral. The services of various Input Method engines are offered across platform boundary through these IIIM Protocol (IIIMP for short). Now the IIIM Servers collectively export Input Method services of different language engines to the IIIM Client which are ultimately used by the client applications.

Writing texts using engine based Input Methods is quite an involved task. These Input Methods require event/message handling structure to process complex languages (e.g CJK). The load on the server and ultimately on the engine is quite high, which makes the processing (of requests) quite slow and practically unusable on the distributed environment. The IIIM Framework deals with this through its various load dispersion mechanisms, which delegate this non-trivial responsibility to the different components in the framework.

We will discuss some of the definitions used in the IIIM Framework and the role played by each of them one by one.

IIIM Protocol (IIIMP): (Note: Please read IIIMP Specification for details) IIIMP supports multiple IM infrastructure. IIIMP is considered to be a platform independent, windowing system independent, and language independent Input Method protocol.

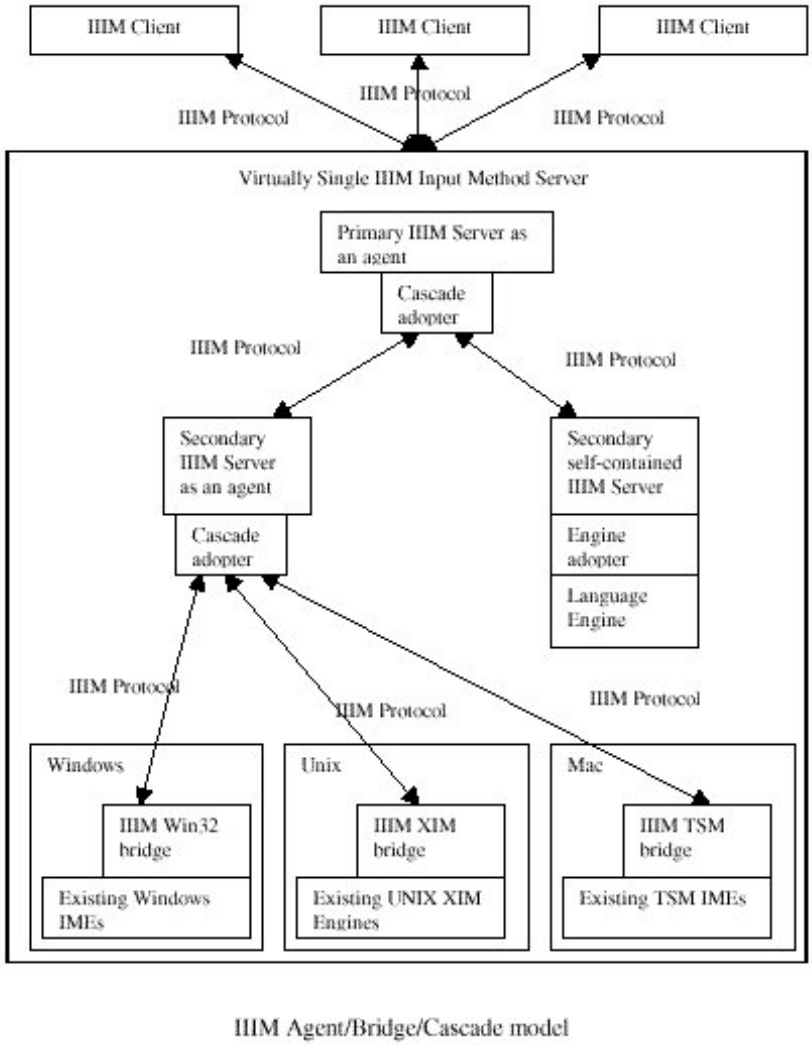
IIIM Client Framework (IIIMCF): IIIMF Client framework handles issues of GUI/Windowing aspects involved in the IIIM Framework. The IIIM Client framework is platform specific. The currently available IIIM Client framework are listed below:-

1. IIIMXCF : IIIM X Window System Client Framework
2. IIIMJCF : IIIM Java Client Framework
3. IIIMECF : IIIM Emacs Client Framework
4. IIIMGCF : IIIM GTK+ Client Framework
5. IIIMWCF : IIIM Windows Client Framework
6. IIIMQCF : IIIM Qt Client Framework
7. libiiimcf : IIIM Generic C Client Framework

IIIM Server Framework (IIIMSF): IIIMF Server Framework is a platform independent Input Method Server platform. IIIMSF provides something called Language Engine Interface Framework (LEIF), which is used by different language engines to interface with the IIIM Server to provide services unlike XIM Servers where each language engine required one to write a separate IM server. These IIIM Servers run as daemons on Linux systems and unlike XIM Server can provide services in different languages simultaneously.

Agent/Bridge/Cascade Model:

Figure 9-2. Agent/Bridge/Cascade Model



IIIM Agent/Bridge/Cascade model

The above diagram gives a generic overview of the Agent/Bridge/Cascade Model of IIIM Framework. In this model each individual IIIM Server is the Agent, there is a Bridge which exports the platform specific input methods to the IIIM Server. The Bridge uses IIIM Protocol to export the input method service to the IIIM Server which receives it through the Cascade Adapter. Now the IIIM Server also acting as the agent can export multiple input method services from multiple platforms at the same time. Multiple levels of agents can exist, each delegating some of its responsibility on the other. This makes the client as thin, so that they are free from complex issues like resource management. IIIM Server appear as a single virtual server to the clients. Also the IIIM protocol has been designed from scratch and abstains itself from using any proprietary frameworks like Java RMI and OMG CORBA. Java has not been used because of the performance factors.

9.6.3. The IIIMF SDK structure

The IIIMF sdk, generally called im-sdk is a collection of libraries which contain many sub-packages to realise the IIIM Framework. Download and unzip the latest version of IIIMF im-sdk (zipped format) from the download section (http://www.openi18n.org/) [IMH1]. At the time this document was written, the name of the latest zipped file was im-sdk-src-r12_1-svn2002.tar.bz2. We will refer to this location as <IIIMSDK_source>. The sub-packages along with their locations are listed below:-

1. IIIMSF: This sub-package is contained in the subdirectory <IIIMSDK_source>/iiimsf/
2. Language Engines: This sub-package is contained in the subdirectory <IIIMSDK_source>/leif/.
3. libiiimcf: This sub-package is contained in the subdirectory <IIIMSDK_source>/lib/iiimcf/.
4. libiiimxcf: This sub-package is contained in the subdirectory <IIIMSDK_source>/iiimxcf/. This client framework packages consist of xiiimp.so library and htt_xbe adaptor.
5. libiiimjcf: This sub-package is contained in the subdirectory <IIIMSDK_source>/iiimjcf/.
6. EIMIL: This sub-package is contained in the subdirectory <IIIMSDK_source>/lib/EIMIL/.
7. CSConv: This sub-package is contained in the subdirectory <IIIMSDK_source>/lib/CSConv/.
8. IIIMGCF: This sub-package is contained in the subdirectory <IIIMSDK_source>/iiimgcf/.

9.6.4. How-To use IIIM

To run and use IIIM for GTK+ or QT based applications follow the following steps:-

1. Start the IIIMF server by typing the command

   ```
   /usr/lib/im/htt &
   ```

   or by simply typing

   ```
   htt_server
   ```

2. Type the command

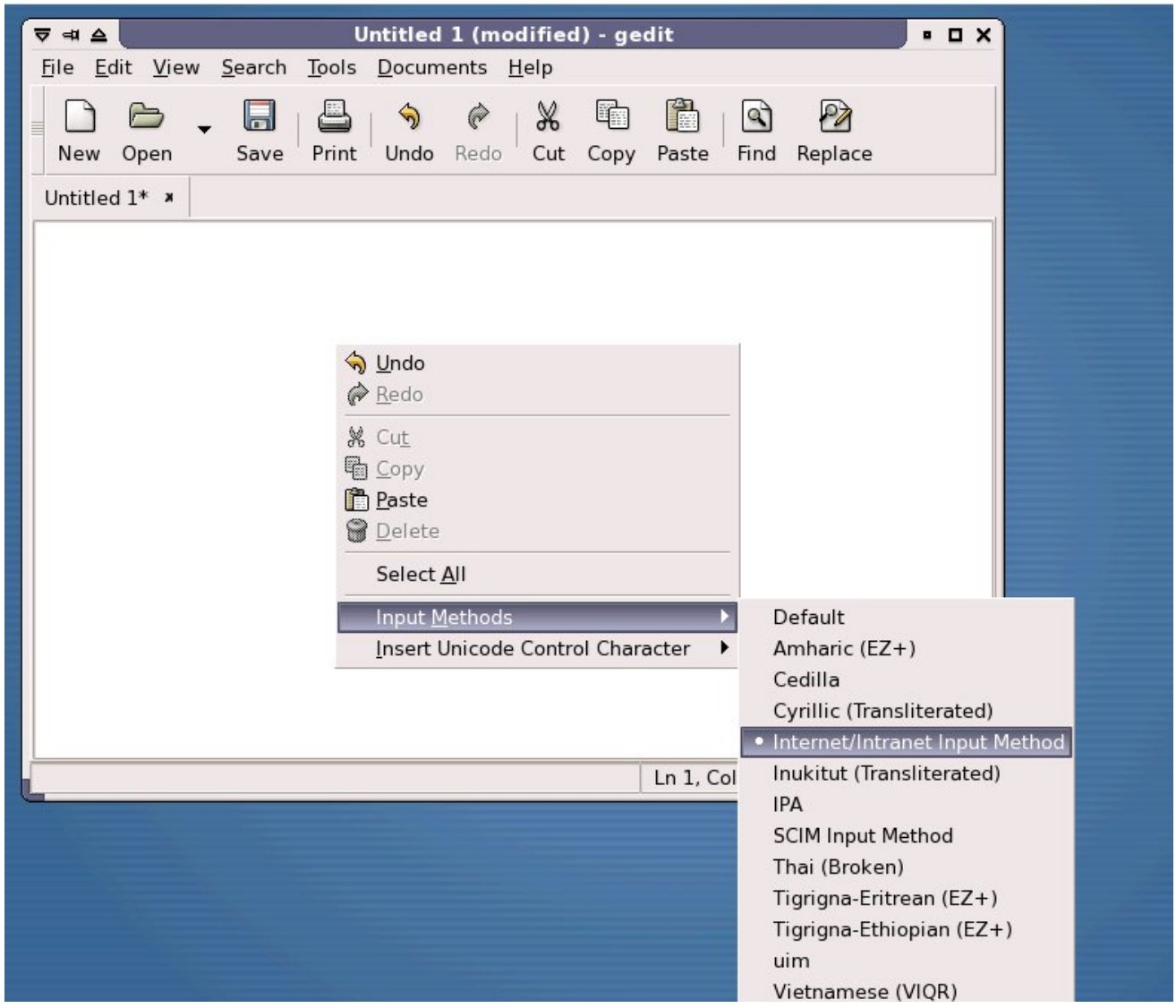
   ```
   export GTK_IM_MODULE=iiim
   ```

   for GTK+ or

```
XMODIFIER=@im=htt
```

for Qt.

This will select iiim as the default Input method Server. You can also select the IIIMF by right clicking on the application and selecting the appropriate input method as shown in the figure below.



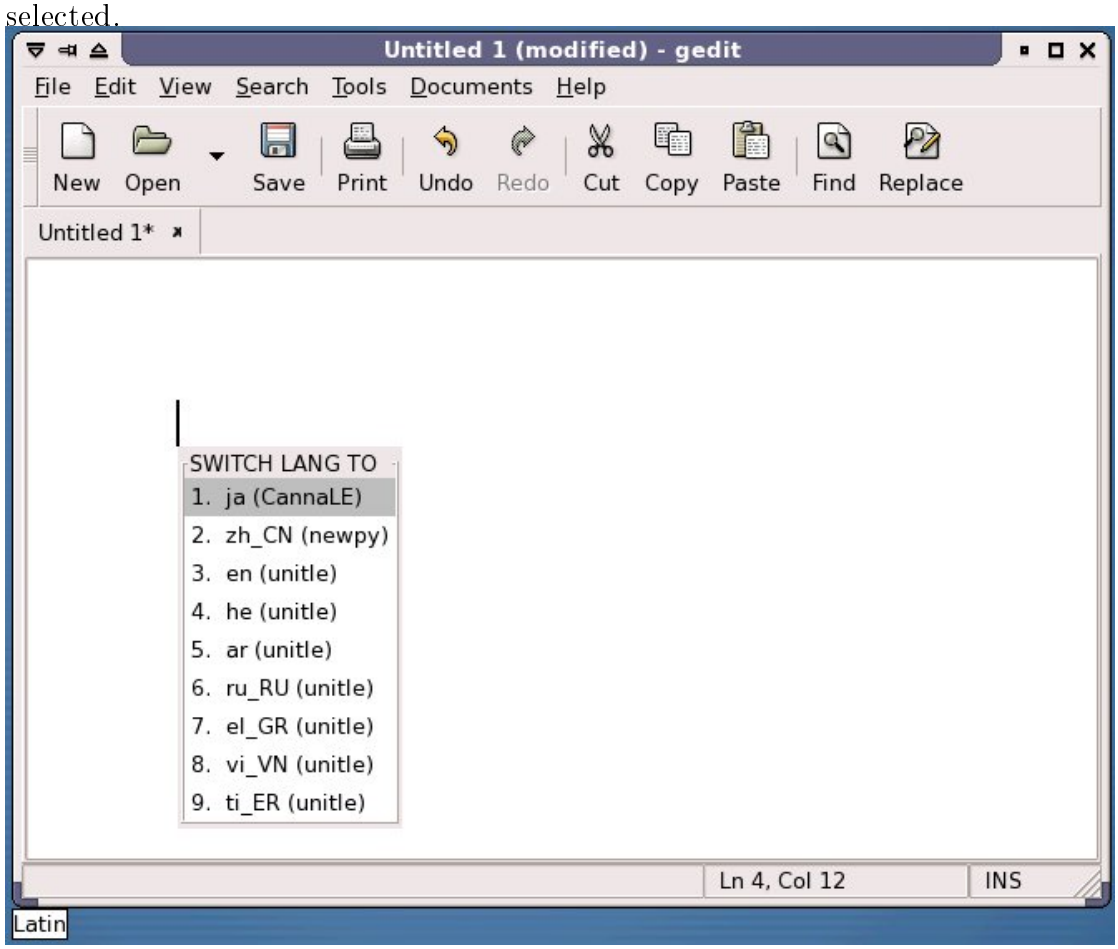
3. Select the locale of your choice.

   For e.g.

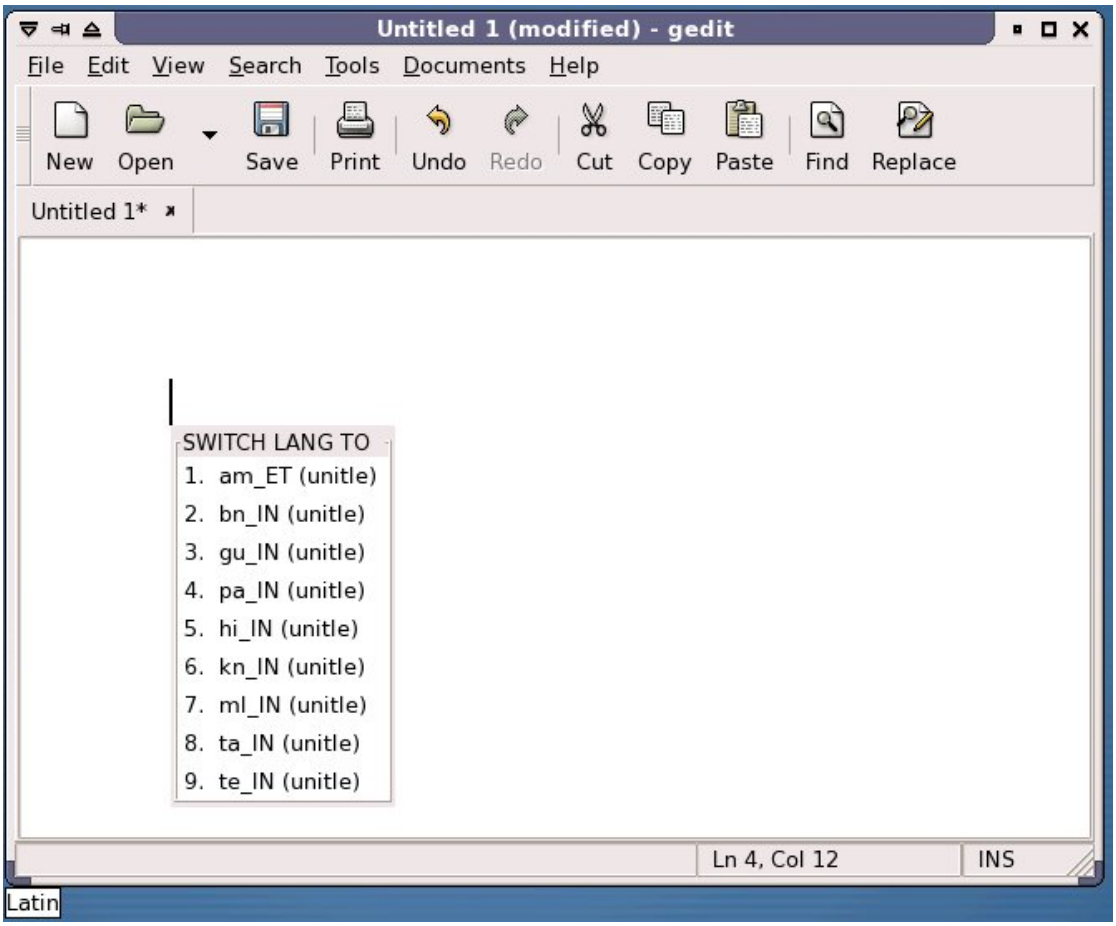
```
export LANG=hi_IN.UTF-8
```

4. Hold The CTRL Key and Press SPACEBAR (CTRL+SPACE). This will activate the IIIM input method which is confirmed by the presence of a status area window like
   Latin
   in the lower left corner of the application window.

   (see screen below)Then press CTRL, ALT and L keys (CTRL+ALT+L).This will open an Auxiliary Window area which contains a list of the languages that can be selected.



5. Scroll among these languages using the keys PAGE UP and PAGE DOWN and select any language in which you want to enter text.



6. The text in status area window changes to notify the change of language.

   For e.g

   Latin (en) to Hindi(hi_IN) the status area changes from

   Latin

   to

   शब्दलिपि

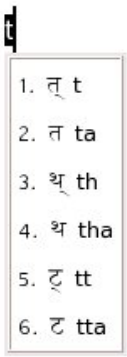
   .

7. Start typing in your language using the different keymaps (e.g Inscript, Transcript, Phonetic methods) available. One can switch between these ways using the key F6 on the keyboard.

9.6.5. A simple Example in Hindi

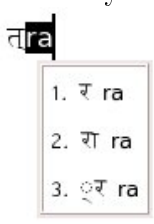
After we have followed the above steps, we can enter the text in our desired language (Hindi in this case). Follow the following steps:

1. Select your desired language (explained above). In this example we have selected Hindi (hi_IN).
2. Select your keymap from amongst available. We selected transcript in this case which is depicted as
   शब्दलिपि
   on the status area. Note that Inscript form is depicted a
   इंस्क्रिप्ट
3. Suppose we want to type the Hindi letter
   त्र
   which can be typed only by a combination of other letters. If we transliterate this alphabet using English phonetic letters we express this with letters <tra>. This can be divided into two parts as half <ta> and <ra>. This half<ta>, represented as <t>, can be created using <ta> and a graphical sign known as halant in hindi. This formula is shown below.
   त + ् + र = त्र
   We observe that the letter has been transformed into a shape different from the original combination.
4. To enter
   त्र
   we start by typing <t>. We are presented with a list of choices, in a pop-up auxiliary window, as shown in the figure below.



   We select 1st option which is half <ta>.

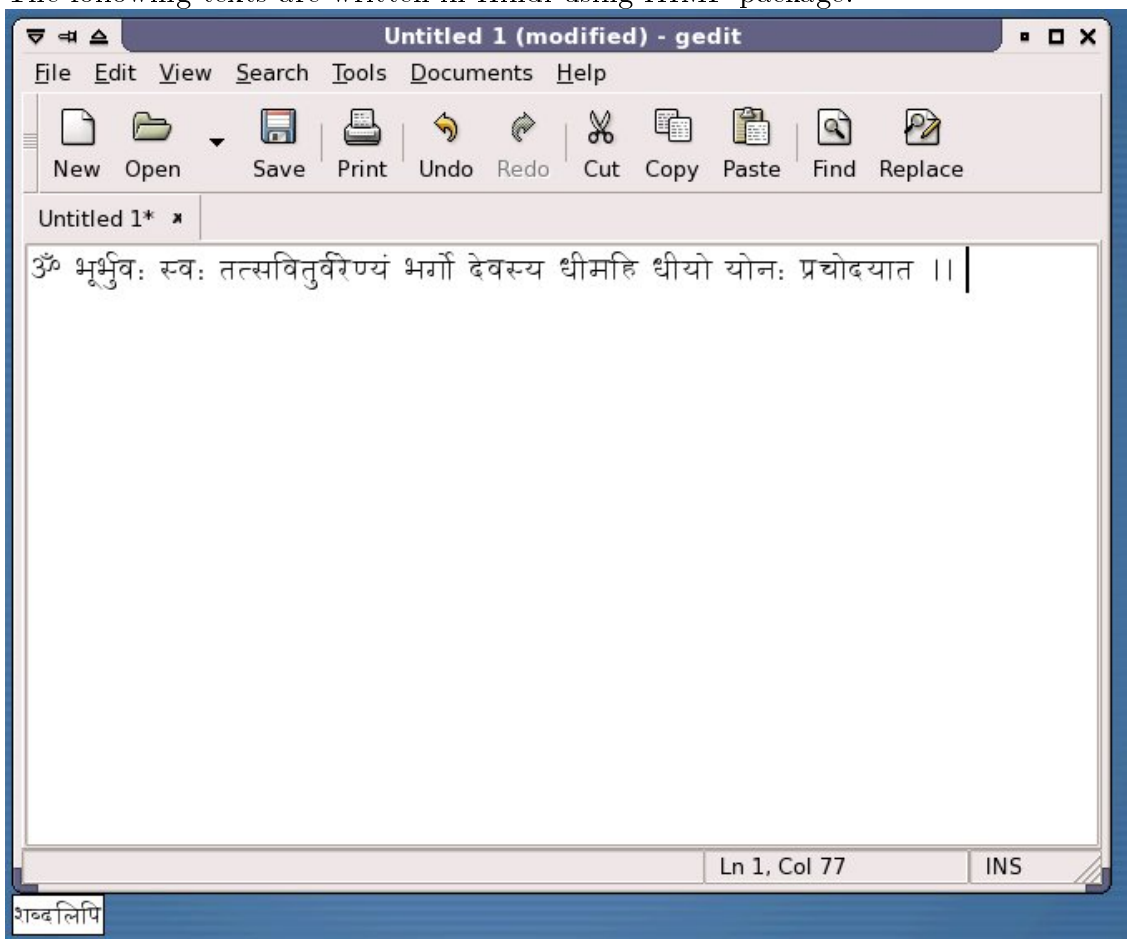
   Next we type <ra>; we are again presented with a list of choices, in a pop-up auxiliary window,as shown in the figure below.



   Again we select the 1st option, which is <ra>. Note that letters <ra> in black region, shown above, is the pre-edit region.

In pre-edit region, we can edit the letters before selecting a letter (in desired language) from the auxillary window area. One point that must be remembered is that once we have committed the selection, it cannot be re-edited and only option left with us is to delete the full letter and retype it.

5. The following texts are written in Hindi using IIIMF package.



9.6.6. Case Study of UNIT language engine

In this section we describe UNIT language engine. UNIT stands for UNIcode Table based Input Method. UNIT is a generic multilingual language engine for those languages which can be input using generic table lookup approach.

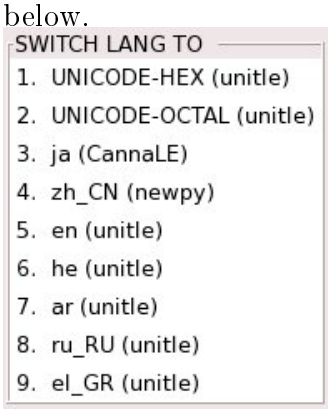
(Note: Please locate the home directory of iiim. This is usually located in the directory /usr/lib/im/, henceforth we will refer to this directory as <IIIMF_HOME>). For extensive information on UNIT refer to [TIM16].

The languages which are supported at the time this text is written are:

1. **Hindi**
2. **Bengali**
3. **Gujarati**

4. **Malayalam**
5. **Gurumukhi**
6. **Tamil**
7. **Telugu**
8. **Cyrillic**
9. **Hebrew**
10. **Arabic**
11. **Greek**
12. **Vietnamese**

Apart from these languages, Unicode Codepoint based input methods are also supported. This means we can theoretically enter any letter for which UNICODE Codepoint is defined. This selection can be made from the auxillary window as shown below.



We can add and delete UNIT languages, and even keyboard layouts, from the run time language auxillary window by making appropriate changes (commenting or deleting languages or keyboard layouts) in the file <IIIMF_HOME>/locale/UNIT/sysime.cfg. A view of this file is shown below.

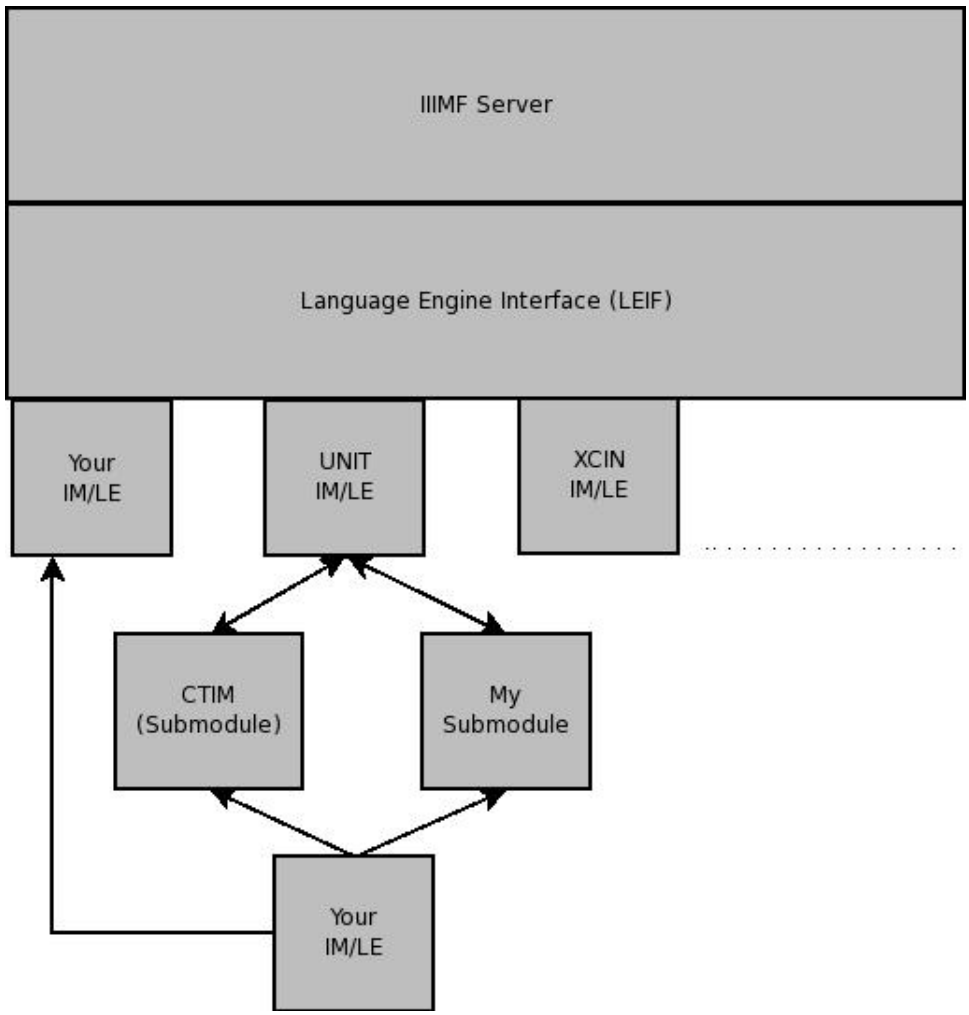
sysime.cfg

```
[ GENERIC_IM_TABLE ]

[ SWITCH_LOCALE ]
IM_VK_F5 0

[ SWITCH_LAYOUT ]
IM_VK_F6 0

[ en ]
euro common/xctim.so EUROPEAN

[ he ]
hebrew common/xctim.so HEBREW

[ ar ]
arabic common/xctim.so ARABIC
```

```
[ ru_RU ]
cyrillic common/xctim.so CYRILLIC

[ el_GR ]
greek common/xctim.so GREEK

[ vi_VN ]
vietnamese common/xctim.so VIETNAMESE

[ ti_ER ]
eritrean common/xctim.so TI_ER

[ am_ET ]
amharic common/xctim.so AM_ET

[ bn_IN ]
inscript common/ctim.so BENGALI
#phonetic common/phonetic_im.so BENGALI

[ gu_IN ]
inscript common/ctim.so GUJARATI
#phonetic common/phonetic_im.so GUJARATI

[ hi_IN ]
trans common/ctim.so HINDI
inscript common/ctim.so HINDI
#phonetic common/phonetic_im.so HINDI

[ kn_IN ]
inscript common/ctim.so KANNADA
kgp common/ctim.so KANNADA
#phonetic common/phonetic_im.so KANNADA

[ ml_IN ]
inscript common/ctim.so MALAYALAM
#phonetic common/phonetic_im.so MALAYALAM

[ ta_IN ]
inscript common/ctim.so TAMIL
trans common/ctim.so TAMIL
#phonetic common/phonetic_im.so TAMIL

[ te_IN ]
inscript common/ctim.so TELUGU
#phonetic common/phonetic_im.so TELUGU

[ UNICODE-HEX ]
```

```
codepoint common/codepoint_im.so UNICODE-HEX

[ UNICODE-OCTAL ]
codepoint common/codepoint_im.so UNICODE-OCTAL
```

9.6.6.1. How to create your own Input Methods



The above diagrams explains the abstract view of the language engine framework with respect to IIIMF. You can add your language's input method or language engine to UNIT Framework either as a part of Codetable Based Input Method (CTIM) or as a submodule to UNIT.

1. To create an intelligent input method, write your own submodule, and create a library out of it(shared or static).
2. This library must be installed under the directory <IIIMF_HOME>/locale/UNIT/common/
3. Now add the entry for this file in the file <IIIMF_HOME>/locale/UNIT/sysime.cfg. The structure of this file was shown

above.

9.6.6.1.1. Creating your own CodeTable

1. Select your keyboard layout and create a file like shown below. This is a sample file called trans.utf, which is a transcript layout for Hindi language. This file is located under <IIIMSDK_source>/leif/unit/dict/HINDI/

```
## HINDI codetable input table

[ Description ]
Locale Name: हिंदी
Layout Name: शब्दलिपि
Encode:              UTF-8
UsedCodes:    abcdefghijklmnopqrstuvwxyzADGIJKLNORTU1234567890!
@#$%^&*()_-+=|\~`{[}]:;"'<,>.?/
MaxCodes:     5

[ Function_Key ]
PageUp:              [< ^P
PageDown:     ]> ^N
BackSpace:    ^H ^?
[ Options ]
KeyByKey_Mode:              ON
HelpInfo_Mode:              ON
AutoSelect_Mode:     ON
KeyPrompt_Mode:             ON
SelectKey_Mode:             Number

[ Single ]
! !
1 १
@ @
2 २
# #
3 ३
$ $
```

Now in order to save the pages the rest of the file is listed in the table below (read from left column to right column).

Table 9-3. A keymap for Hindi

4 ४	+ +	, ,	aa आ ा	kh ख्	nga ङ	dda ड	bh भ्	v व्	da द	
% %	= =	> >	A आ ा	kha ख	nya ञ	Dh ढ़	bha भ	va व	dh ध्	
5 ५	| |	. .	ae ए े	g ग्	jna ज्ञ	Dha ढ	bhaa भा	w व्	dha ध	
^ ^	\ \	? ?	ai ऐ ै	ga ग	gya ज्ञ	ddh ढ़	m म्	wa व	n न् ं	
6 ६	~ ~	/ /	au औ ौ	gh घ्	T ट्	ddha ढ	ma म	s स्	na न	
& &	` `	i इ ि	aM अं	gha घ	Ta ट	N ण्	y य्	sa स	sri श्रि	
7 ७	{ {	I ई ी	am अं	ch च्	tt ट्	Na ण	ya य	sh श् ष्	srI श्री	
* *	[[	u उ ु	aH अः	cha च	tta ट	nn ण्	r र्	sha श ष	sree श्री	
8 ८	} }	U ऊ ू	ah अः	chh छ	Th ठ	nna ण	ra र	ha ह	shri श्रि	Ka ख़
((	]]	e ऐ	uu ऊ ू	chha छ	Tha ठ	p प्	l ल्	ksha क्ष	shree श्री	Ga ग़
9 ९	: :	o ओ ो	ru ऋ ृ	j ज्	tth ठ	pa प	la ल	t त्	ra रा ्र	Ja ज़
))	; ;	R ॠ	Ru ॠ ॄ	ja ज	ttha ठ	ph फ्	lla ळ	ta त	om ॐ	za ज़
0 ०	" "		RRu ॡ	jh झ्	D ड़	pha फ	La ळ	th थ्	OM ॐ	fa फ़
_ _	' '	[Phras	k क्	jha झ	Da ड	b ब्	L ॢ	tha थ	AUM ॐ	Da ड़
- -	< <	a अ	ka क	ng ँ	dd ड़	ba ब	LL ॡ ॣ	d द्	qa क़	Dha ढ़

This file can be used to add more characters based on keymap (i.e for each combination of letters in English, corresponding glyph in Hindi can be specified.). All you need to do is to type letters in English followed by a tab followed by Hindi letters.

2. Copy this file to the directory <IIIMF_HOME>/locale/UNIT/common/. Convert this file to binary format using the command txt2bin, which is located under the same directory by using the command.

```
txt2bin<source_file> <binary_file>
```

3. Add keyboard layout for your language in the file <IIIMF_HOME>/locale/UNIT/sysime.cfg
4. Copy the file created in step 1 to <IIIMF_HOME>/locale/UNIT/<YOUR LANGUAGE NAME>/data/ directory. For example for Hindi this will be copied to <IIIMF_HOME>/locale/UNIT/HINDI/data/.

Chapter 10. Gettext: Architectural Overview

The word Localisation (L10N) can not be explained without mention of the word Internationalisation (I18N). When we say that a particular software is Localisable, we imply that this software is already adequately Internationalised, which semantically means that the application/software can be easily ported/adopted to a particular language. The software/application have been generalised and there are standard procedures to adapt it to a particular language. The software can be Localised in any language, of users choice, by using these standard approaches/procedures.

Gettext framework is one such approach. It refers to a collection of tools which are used to translate messages and Strings output by an application. These tools assist in translating the strings on menus, messages boxes or icons on the applications in the language that the user is interested in. Although because of the complexity of the translation procedure a naive user cannot do this kind of translation by himself. The translation of this application is done by an expert team on behalf of the user community who use this software.

The first step in doing translation of strings and messages of an application is to identify those strings and messages which needs to be translated. After such an identification these strings must be extracted into some file. This is done in order to hand it over to translators who can independently work on the translation of the strings in these files. Translators create corresponding translations for the strings in the same file , in the desired language. When this file is handed over to the translators, the corresponding translations (in target language) are empty and these serve as template files for the translators. In gettext framework these files are given the extension of .pot (we will simply call these POT files), POT stands for portable object template.

Because any application evolves in stages, the strings and messages of that application change and add-up over a period of time and so do these POT files. This temporal nature of POT files make it ineffective or unpractical to contain the translations in the same file. Also we can't modify the POT files because the software might need to be translated into more than one language (hence template for translation not oriented towards any particular translation).

One solution here is to finish the development of an application first, then create the POT files, then hand it over to the translators. But this proves to be an expensive option, because by the time, translation for that language is complete, the software might be obsolete, or new versions of software might be out.

The other solution is to translate the strings (in a particular language) in another file in parallel to the development of application. The format of these new files must be the same as that of the POT files. Then there must be some mechanism through which the updated strings in POT files are merged with the strings in this new file without any inconsistencies. The corresponding translations (in any particular language) of these updated strings are empty, and now the translators can work on these strings along

with other untranslated strings. These new files are given the extension .po (we will simply call it PO files). PO files initially look the same as the POT files when they are handed over to the translators(without any translation).

Now translators work on these PO files and translate each string one-by-one. The original string in these files are represented by the keyword msgid and strings are kept between a pair of "". The corresponding translations are represented by the keyword msgstr with translated strings in between a pair of "" in PO files and as empty strings in POT files between a pair of "". The format of PO and POT files will be explained in detail later.

Now the translation and development can go in parallel. POT files are continuously updated with new strings, these strings are merged with the PO files on which the translators are working. Hence translators are fed with additional strings to translate each time the Files are merged. The development and localisation of the software can go in sync with each other.

After it is felt that a stable version of translation is ready the files can be merged back to the application. Since there are multiple languages which the software can support, there must be multiple glossary/file from which the translations can be extracted. In gettext framework, these files are given the extension .mo (we will simply call them MO files). These MO files are binary files which are linked with the applications at run time and provide text for display in the desired language. The format of MO file is not portable across the platforms and hence are platform specific. The GNU gettext format for the MO file has the extension .gmo.

We will explain all these in detail as the text progresses.

10.1. The POT and PO file format

Let us now discuss the format in which the POT and PO files are written. The previous section has explained the origin and significance of these files. The POT files and PO files have format similar to each other . The difference lies only in the filename extensions and intended usage.

There are editors/IDE which abstract translators from the format of PO and POT files, and help in the translation process. The extraction of strings from the source files is handled by the IDE itself. Also these editors/IDE take care of the versioning aspects. This mode in which these editors/IDEs operate has been given a special name as the PO-Mode. There are several freely available PO-Mode editors/IDE which help in the translation process. Some examples of such editors/IDE are Yudit, gtranslator, KBabel, emacs etc.

The general structure of PO file is given in the table below.

```
white-space
#  translator-comments
```

```
#. automatic-comments
#: reference...
#, flag...
msgid untranslated-string
msgstr translated-string

#: reference...
#, flag...
msgid untranslated-string
msgstr translated-string
```

The entry in the PO files begins with the optional white space in the beginning. The comments begin with a #. The comments where the character # is followed by some other special character such as '.'(dot) and ';'(semi-colon) etc are inserted automatically by the gettext program during the extraction process. And the comments where the character # is followed by a whitespace are inserted by the translator who is translating the strings in the PO files. Both types of comments are optional. The comment where the character # is followed by a ,(comma) contains a ,(comma) separated list of flags, which specify the format of the translated string. This flag format will be explained shortly.

Then there are a series of lines in pair starting with the keywords msgid and msgstr respectively. In the above table msgid is shown first followed by a string in the source language, followed by a msgstr in the next line which is immediately followed by a blank string in POT file and translated string in the PO file. One example of this is given in the table below.

```
# Some comment
#
msgid ""
msgstr ""
"Project-Id-Version: MyPackage 1.0\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2005-01-12 01:51+0530\n"
"PO-Revision-Date: 2005-01-12 01:52+0530\n"
"Last-Translator: Naveen Kumar <nav007@gmail.com>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit"

#: some comment
#, c-format
msgid "Failed to open file : %s"
msgstr "translation in your language %s"
```

```
#: some comment
#, c-format
msgid "image file %s contains no data"
msgstr "translation in your language %s"
```

When translating to Hindi, the same file would look like the following:

```
# Some comment
#
msgid ""
msgstr ""
"Project-Id-Version: MyPackage 1.0\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2005-01-12 01:51+0530\n"
"PO-Revision-Date: 2005-01-12 01:52+0530\n"
"Last-Translator: Naveen Kumar <nav007@gmail.com>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"

#: file.c:16
#, c-format
msgid "Failed to open file : %s"
msgstr "फाइल %s खोलने मे असमर्थ"

#: file.c:18
#, c-format
msgid "image file %s conatins no data"
msgstr "छवि फाइल %s में कोई आंकड़ा नहीं है"
```

The flag format (mentioned earlier in the program) can be one of the following types:

fuzzy : These entries are created when the msgmerge command is used to update the PO files with the newly introduced strings in the corresponding POT files. The new string might introduce some ambiguity by introducing a text in PO files which are only a slight modification of an earlier text in the PO file. The ambiguity is that with such slight modification, the translation may or may not change. The only person who can decide on this is the translator himself. The msgmerge program marks such ambiguous entries as fuzzy. Now the translator can look at these fuzzy strings and then decide whether to change the translation or not. He can then remove this fuzzy attribute.

c-format : This entry is added by the xgettext program. This means msgid part of the string is written in a format similar to the printf function (in C-programming language). This also means that the translator must ensure that the format of the translated string is similar to the format (c-format) of the original string in msgid. The msgfmt program is used to check the validity of the string.

no-c-format : This entry is also added by the gettext program.

For more information on PO file format, refer to [POT1].

10.2. Internationalisation

We can categorise Software Internationalisation in three ways.

- Compile Time Internationalisation
- Link Time Internationalisation
- Run Time Internationalisation

We will explain the internationalisation process with the help of a C language program. The target language in which we will translate the program is Hindi(Devanagari script). The original source code for the program is given below

```
#include<stdio.h>
int main()
{
      int i;
      printf("again");
      printf("\n");
      printf("Hello World");
      printf("\n");
      printf("These text demonstrate that translation using po files
                                                          is easier\n");
      scanf("%d",&i);
     if(i==1)
              printf("This number %d is of singular form");
     else
              printf("This number %d is of plural form");
      printf("\n");
      printf("The magic number is %d\n",i);
      printf("These are additional words");
      printf("\n");
      printf("I love translation\n");
}//main
```

10.2.1. Compile Time Internationalisation

The above program is monolithic and is not Internationalised. If someone wants to localise this program into his own language, he has to refine and rewrite the whole source. Internationalisation process provides an alternative.

The above program can be modified to facilitate for the Compile-Time Internationalisation. The source is modified as alternative shown below(helloworldcomt.c).

Figure 10-1. helloworldcomt.c

```
#include<stdio.h>
#ifndef LANGUAGE
#define LANGUAGE 1
#endif

#ifndef en
#define en 1
#endif

#ifndef hi
#define hi 2
#endif
int main()
{
    int i;
    #if LANGUAGE==en
        printf("again");
        printf("\n");
        printf("Hello World");
        printf("\n");
        printf("These text demonstrate that translation using po files is easier\n");
        scanf("%d",&i);
            if(i==1) printf("This number %d is of singular form",i);
            else printf("This number %d is of plural form",i);
            printf("\n");
            printf("The magic number is %d\n",i);
            printf("These are additional words");
            printf("\n");
            printf("I love translation\n");
    #elif LANGUAGE==hi
        printf("फिर से");
        printf("\n");
        printf("नमस्कार दुनिया");
        printf("\n");
        printf("ये शब्द ये दर्शाते हैं कि po फाइलों के द्वारा अनुवाद करना आसान है\n");
        scanf("%d",&i);
            if(i==1)
                printf("%d अंक एकवचन है",i);
            else
                printf("%d अंक बहुवचन है",i);
            printf("\n");
            printf("चमत्कारी अंक %d है\n",i);
            printf("ये अतिरिक्त शब्द हैं");
            printf("\n");
            printf("मुझे अनुवाद करना पसन्द है\n");
    #endif
}
```

Now this program can be compiled separately for English and Hindi.

For English it is compiled in the following manner

```
prompt# gcc -D LANGUAGE=en -o helloworldcomt helloworldcomt.c
```

For Hindi it is compiled in the following manner

```
prompt# gcc -D LANGUAGE=hi -o helloworldcomt helloworldcomt.c
```

We get different executable files for each language. One must use the appropriate one depending on which language one wants to use.

The Compile Time Internationalisation takes place during the compile time of the program. In a way this program can be Internationalised but its still monolithic, as far as different languages are concerned. This form of internationalisation is a kind of burden on the programmer as the original source needs to be refined and recompiled for the addition of each language for which localisation is desired. Besides this slows down the overall software development process. As it is said the localisation process should not be a burden on the developers but the community which wants to localise, a generic internationalisation framework is required, which facilitates localisation without the source code change and recompilation.

10.2.2. Link Time Internationalisation

The problem mentioned in the last section can be dealt with Link Time Internationalisation of our program to some extent. We prepare different libraries specific to different locales(languages). The library specific to a particular locale(language) is linked with the main application to produce an executable for that locale.

Now the above program is modified and split into three programs to enable us to utilise the power of Link Time Internationalisation. The idea is to separate the parts which are likely to be modified for localisation from the rest. For these parts, we can create different versions, leaving a fair part of the full source code unaffected.

helloworldlinkt.c

```
#include<stdio.h>

extern void helloworld();

int main()
{
        helloworld();
}//main
```

Figure 10-2. helloworldlinkt_hi.c

```
#include<stdio.h>

int helloworld()
{
    int i;
        printf("फिर से");
        printf("\n");
        printf("नमस्कार दुनिया");
        printf("\n");
        printf("ये शब्द ये दर्शाते हैं कि po फाइलों के द्वारा अनुवाद करना आसान है\n");
        scanf("%d",&i);
            if(i==1)
                printf("%d अंक एकवचन है",i);
            else
                printf("%d अंक बहुवचन है",i);
            printf("\n");
            printf("चमत्कारी अंक %d है\n",i);
            printf("ये अतिरिक्त शब्द हैं");
            printf("\n");
            printf("मुझे अनुवाद करना पसन्द है\n");
}
```

helloworldlinkt_en.c

```
#include<stdio.h>


void helloworld()
{
      int i;

      printf("again");
      printf("\n");
      printf("Hello World");
      printf("\n");
      printf("These text demonstrate that translation using po files
                                                              is easier\n");
      scanf("%d",&i);
    if(i==1)
            printf("This number %d is of singular form");
           else
            printf("This number %d is of plural form");
      printf("\n");
      printf("The magic number is %d\n",i);
      printf("These are additional words");
      printf("\n");
      printf("I love translation\n");
```

```
}//main
```

These programs are compiled to create object files in the following ways.

```
prompt# gcc -c -o helloworldlinkt.o helloworldlinkt.c
prompt# gcc -c -o helloworldlinkt_hi.o helloworldlinkt_hi.c
prompt# gcc -c -o helloworldlinkt_en.o helloworldlinkt_en.c
```

Now the individual language packages are converted into corresponding libraries using the following commands.

```
prompt# ar rcs libhelloworldlinkt_hi.a helloworldlinkt_hi.o
prompt# ar rcs libhelloworldlinkt_en.a helloworldlinkt_en.o
```

To get more information on how to create libraries please refer to [PLH1].

After the libraries have been created, they are copied to the /usr/lib/ directory.

Now as per requirements, these languages are linked with the main object file to create a corresponding executable for the required language.

```
prompt# gcc -o helloworldlinkt helloworldlinkt.o -lhelloworldlinkt_en
prompt# gcc -o helloworldlinkt helloworldlinkt.o -lhelloworldlinkt_hi
```

This Internationalisation solution still has some disadvantages. We need to create a separate executable for each language, and if there is a need we cannot switch between these languages at run time.

Clearly the solution we need is having one executable for the main application and having corresponding translations (for each language) in different files/database tables, which are resolved at run-time to give the localised version of the application.

10.2.3. Run Time Internationalisation

The problems that we were discussing in the previous section can be resolved using Run-Time Internationalisation of the software. The trick here is to have a single executable and let the user select a locale at run-time. These locales are mapped to a particular language translation at run-time which presents, to the users, the localised version of the application. These translations can be kept in a database/glossary in particular format as an object code, which are linked at run-time depending on the locale.

Major advantage of this kind of localisation is that it separates the source code from the language translations, which in turn, has advantages from two perspectives:

- It frees the programmer from the Localisation related aspects and he concentrates solely on the application development.
- It frees translators from the source code aspects and they can solely concentrate on the translation related issues.

gettext has the framework to facilitate Run-Time Internationalisation of the software. In the next section, we will see how this is achieved.

10.3. Localisation through gettext

Almost 90% applications today on Linux platform are localised using the gettext framework. Internationalisation and localisation process through gettext framework is achieved through the following phases, explained in detail in the subsequent subsections below.

- Preparing the source code for internationalisation
- Extraction process
- Translation process
- Compilation of translation
- Retrieval of translation

10.3.1. Preparing the source code for internationalisation

Internationalisation in Linux is achieved with the help of libintl library. Libintl (part of gettext) provides native language support to programs. Because libintl is included with GNU libc library, we do not need to link against either libintl.a or libintl.so. The only requirement for our programs to use libintl is to include libintl.h header file.

Now in order to use the internationalisation framework we need to do some modifications in our original source code. The code after these modifications is shown below.

helloworldintl.c

```
#include<libintl.h>
#include<locale.h>
#include<stdio.h>

#define _(String) gettext (String)
#define _t(String1,String2,n) ngettext (String1,String2,n)

int main()
{
    int i;
    setlocale(LC_ALL,"");
    bindtextdomain("helloworldintl","/usr/share/locale");
    textdomain("helloworldintl");
    printf(_("again"));
    printf("\n");
    printf(_("Hello World"));
```

```
    printf("\n");
    printf(_("These text demonstrate that translation using po files
                                                            is easier\n"));
    scanf("%d",&i);
    printf(_t("This number %d is of singular form","This number %d
                                                  is of plural form",i),i);
    printf("\n");
    printf(_("The magic number is %d\n"),i);
    printf(_("These are additional words"));
    printf("\n");
    printf(_("I love translation\n"));
}//main
```

This example code makes use of the functions gettext and ngettext (explained below in detail). These functions are used to mark the strings, that should be localised/translated, to prepare the code for internationalisation. The macros _("") and _t("") are used, instead of original functions, to reduce the number of additional characters to three, per string that needs to be translated. This is a common practice among the application developers using gettext framework to internationalise their application. The above code now can be said to be fairly internationalised. The most common functions are summarised below.

Extraction functions

- gettext
- dgettext
- ngettext
- dngettext
- dcngettext

Other functions

- textdomain
- bindtextdomain
- setlocale

The extraction functions (listed above) are used to mark the strings in different ways, so that they can be extracted during the extraction phase. The three functions textdomain, bindtextdomain and setlocale help the executable (compiled version of the application) in deciding from where to select translated string at run-time.

The above program uses four of these mentioned functions; the others are similar with little variations. Hence, we will explain these four in terms of their significance in the program.

The functions gettext and ngettext are used by the program xgettext to extract strings (which need to be translated), and put them in the POT file(s). While the first function has only one parameter (the string to be extracted), the second one has three parameters. The second function (i.e, ngettext) is used in order to get translation for the plural forms. The functions bindtextdomain and textdomain are used to specify the domain of the message catalog, from where the executable will get their translations based on their locales (if the translations are present for the locale). The first function may be omitted but the second function textdomain is necessary for the proper working of translation procedure. The second function is necessary for the case where we do not want to use the systems default message catalog. It is recommended that we use this function.The setlocale function sets the appropriate value for a particular locale.

10.3.2. Extraction Process

All the strings are extracted in this phase using the extraction functions, mentioned above, and put inside a POT file. The program xgettext is used for the extraction process. For the above source file the extraction is done using the following command.

```
prompt# xgettext -d helloworldintl -o helloworldintl.pot  --keyword=_t:1,2 -k_
                                                    -s helloworldintl.c
```

This command generates the following POT file

helloworldintl.pot

```
# SOME DESCRIPTIVE TITLE.
# Copyright (C) YEAR THE PACKAGE'S COPYRIGHT HOLDER
# This file is distributed under the same license as the PACKAGE package.
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
#
#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2004-12-06 04:29+0530\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: 8bit\n"
"Plural-Forms: nplurals=INTEGER; plural=EXPRESSION;\n"

#: helloworldintl.c:16
#, c-format
msgid "Hello World"
msgstr ""
```

```
#: helloworldintl.c:30
#, c-format
msgid "I love translation\n"
msgstr ""

#: helloworldintl.c:27
#, c-format
msgid "The magic number is %d\n"
msgstr ""

#: helloworldintl.c:28
#, c-format
msgid "These are additional words"
msgstr ""

#: helloworldintl.c:22
#, c-format
msgid "These text demonstrate that translation using po files is easier\n"
msgstr ""

#: helloworldintl.c:25
#, c-format
msgid "This number %d is of singular form"
msgid_plural "This number %d is of plural form"
msgstr[0] ""
msgstr[1] ""

#: helloworldintl.c:14
#, c-format
msgid "again"
msgstr ""
```

10.3.3. Translation Process

The translation process starts with the copying of POT files to the corresponding PO files. Now these po files are translated using a Unicode editor in po-mode. Also a group of PO files can be combined to form a po-compodium file using the command msgcat. As the application grows more and more strings appear inside the application which need to be localised. Hence our po files needs to be updated. This can be done using the command msgmerge.

After translation our PO files looks something like as shown below.

Figure 10-3. helloworldintl.po

```
# SOME DESCRIPTIVE TITLE.
# Copyright (C) 2004 C-DAC Mumbai
# This file is distributed under the same license as the MyPackage package.
# Naveen Kumar <nav007@gmail.com>, 2004.
#
msgid ""
msgstr ""
"Project-Id-Version: MyPackage 1.0\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2004-12-06 05:45+0530\n"
"PO-Revision-Date: 2004-12-06 07:30+ZONE\n"
"Last-Translator: Naveen Kumar <nav007@gmail.com>\n"
"Language-Team: C-DAC Mumbai <nav007@gmail.com>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"
"Plural-Forms: nplurals=2; plural=n != 1;\n"

#: helloworldintl.c:16
#, c-format
msgid "Hello World"
msgstr "नमस्कार दुनिया"

#: helloworldintl.c:25
#, c-format
msgid "I love translation\n"
msgstr "मुझे अनुवाद करना पसन्द है\n"

#: helloworldintl.c:22
#, c-format
msgid "The magic number is %d\n"
msgstr "चमत्कारी अंक %d है\n"

#: helloworldintl.c:23
#, c-format
msgid "These are additional words"
msgstr "ये अतिरिक्त शब्द हैं"

#: helloworldintl.c:18
#, c-format
msgid "These text demostrate that translation using po files is easier\n"
msgstr "ये शब्द ये दर्शाते हैं कि po फाइलों के द्वारा अनुवाद करना आसान है\n"

#: helloworldintl.c:20
#, c-format
msgid "This number %d is of singualar form"
msgid_plural "This number %d is of plural form"
msgstr[0] "%d अंक एकवचन है"
msgstr[1] "%d अंक बहुवचन है"

#: helloworldintl.c:14
#, c-format
msgid "again"
msgstr "फिर से"
```

Now the new messages can be extracted from source file using the usual xgettext command. These messages are merged with the older translated messages in PO file, from the updated POT file, using the command.

```
prompt# msgmerge helloworldintl.po helloworldintl.pot -o helloworldintl.pot
```

10.3.4. Compilation of Translation

After the translation process, if a stable version of translation exists, the PO file can be compiled into a binary MO file format, which is used by the application at run-time to provide the translation. The following command creates a MO file

```
prompt# msgfmt helloworldintl.po -o helloworldintl.mo
```

Now these files are copied to the locale directory for which the translation is to be provided (in this case Hindi). So the locale directory where the helloworldintl.mo is to be copied will be /usr/share/locale/hi/LC_MESSAGES/

10.3.5. Retrieval of translation

In order to retrieve the correct translation at runtime, the corresponding locale LC_MESSAGES is set to that language using the command

```
prompt# export LC_MESSAGES=hi_IN.UTF-8
```

Or the executable can directly be called in combination of this locale.

```
prompt# LC_MESSAGES=hi_IN.UTF-8 ./helloworldintl
```

The program now produces the following output (The user inputs are denoted in bold text).

```
prompt# LC_MESSAGES=hi_IN.UTF-8 ./helloworldintl
फिर से
नमस्कार दुनिया
ये शब्द ये दर्शाते हैं कि po फाइलों के द्वारा अनुवाद करना आसान है
1
1 अंक एकवचन है
चमत्कारी अंक 1 है
ये अतिरिक्त शब्द हैं
मुझे अनुवाद करना पसन्द है

prompt# LC_MESSAGES=hi_IN.UTF-8 ./helloworldintl
फिर से
नमस्कार दुनिया
ये शब्द ये दर्शाते हैं कि po फाइलों के द्वारा अनुवाद करना आसान है
2
2 अंक बहुवचन है
चमत्कारी अंक 2 है
ये अतिरिक्त शब्द हैं
मुझे अनुवाद करना पसन्द है
```

10.3.6. Portability

Some non-trivial question may arise at this stage.

1. How do we port our application to non-GNU systems?
2. How can we upgrade to use the newer version of GNU gettext support in our application/package?
3. Is there a standard structure, which can be followed, to bring homogeneity across all applications being developed? Such a structure will ease the internationalisation and localisation of our applications.

The answer lies within gettext framework itself. There is a program/tool in gettext which does this, called gettextize. gettextize installs or upgrades GNU gettext infrastructure, by preparing a common source package to use gettext. This tool also helps in migration from the previous version of GNU gettext. gettextize copies some common files/directories needed in any application/package which makes use of GNU gettext for internationalisation. Some of these important files and directories with explanation are listed below.

1. A po/ directory is created which contains the PO/translation files for different languages. Initially this directory does not contain any PO/translation file. As the translation teams get interested in your application they provide you with these PO/translation files, for their respective languages, which you will put in this directory (this is possible because your application is fairly structured and internationalised). By default this directory contains other files which are helpful in building/distribution of your application which the localisers need not worry about.
2. An intl/ directory is created when gettextize program is invoked with --intl option. This directory is filled with the files of the intl/ directory of the GNU gettext distribution. intl/ directory contains files which are used to provide internationalisation on systems where GNU libintl is not installed.

The gettextize command is invoked in the following manner.

```
prompt# gettextize -c -f --intl
```

A screen dumb of the files and directories produced after this command is invoked, is given below.

```
total 88
-rw-r--r--  1 root root 53848 2005-04-11 17:19 ABOUT-NLS
-rwxr-xr-x  1 root root 14987 2005-04-11 17:19 config.rpath
-rw-r--r--  1 root root     0 2005-04-11 17:19 configure.in
drwxr-xr-x  2 root root  4096 2005-04-11 17:19 intl
-rwxr-xr-x  1 root root  1988 2005-04-11 17:19 mkinstalldirs
drwxr-xr-x  2 root root  4096 2005-04-11 17:19 po
```

For more information read the gettextize manual, [GTM2] and [GTM3] will also be useful.

Now to structure properly we can even add more directories like doc/ for manuals , src/ for source files, lib/ for library files etc.

More exhaustive resources about gettext can be found at [GTM1].

Chapter 11. Translation

11.1. Translation Guidelines

Translation is the process of accurately rendering a document in another language so that it is suitable for its intended purpose. Translation should be complete, grammatically correct and should be terminology consistent.

Translation is the most important aspect in localisation and is not as trivial as it may sound. Metalanguage works or works that discuss language could be very difficult to translate. Comic texts could also be very difficult to translate. Translating the messages and menus in an application is a heavy budget process and inaccurate translation would result in making the application not useable in that particular language.

Below are some guidelines which we suggest should be followed for a simpler life during the translation process:

- First of all try to find out if someone is already working on your language. If so, try to contact them to get assistance for translating the commonly used terms.
- The translator should be very well acquainted with both the source and target languages.
- The translator should also be aware of the context of translation. For example, in the sentence ' Sita made him a duck ' , the word 'duck' can be translated only if context is understood. Such issues will come up when you are translating multiple paragraphs of texts.
- Join a mailing list. You can use this to discuss translation of difficult words.
- It would be a good idea to create a website to tell people about your work, keep glossaries etc.
- You should make sure that your translation consistently uses the same terms. Maintaining a glossary of terms would ensure that you don't use multiple terms to refer to the same thing. As far as possible avoid creating new terms. Try to find out a standard body for your language to get terms.
- The gettext is a good package with tools for internationalisation, managing different versions of the application etc. If you do not know about gettext you can get more information here.
- Avoid word-to-word translation and try to perform sense-to-sense translation. This means that the translator should always bear in mind the intended meaning in the source language.

- The translator should take care to produce the intended overall effect with the appropriate tone by making the right choice of words.
- Be sure that you do not use terms that are jargons or slang.
- Do not use terms that have several meanings.
- Do not use the same words for different meanings.
- Person and number should be retained wherever possible in translation so that singular does not change to plural and third person statements do not change to first or second person statements.
- In some cases a sentence would be highly compressed in English but would run into 2 or 3 sentences when translated. Such sentences need special attention. In such cases also word-to-word translation would cause problems.
- Be sure that the reference to menus and buttons (like "EDIT") matches the term used in the localised operating system.
- Try to get the translation work reviewed by at least two translators independently.
- It would also be benificial to conduct some usability workshops so that you are aware of whether the translations actually make sense to ordinary users.

11.2. Creating a Glossary of terms

11.2.1. What is a glossary?

In the context of localisation, a glossary is a list of English terms that are used to define software terms and their translations in a foreign language, in our case hypolan. A typical glossary entry includes the English term and its corresponding translation in hypolan. The entry may also include a cross reference to similar or related terms.

What terms should be included in a glossary depends upon your project. You may decide to include only the key software terms or may choose to include every term you translate to avoid different translations for the same term. Ideally, the latter approach should be followed so that the translations are unambiguous.

11.2.2. What terms to include in the glossary

When you are creating a glossary you should include the following things:

- New terms you translate - The English term along with its hypolan translation should be written. Also, if possible, the context of the translation should be given so as to avoid ambiguities for subsequent users.

- Synonyms - Terms with similar meanings should be included in the glossary. These variations would be needed with the change in context.
- Common terms used in a non-standard way - Terms that your audience is familiar with but you are using it in a different way.
- Terms that you are just transliterating and not translating.

11.2.3. Format of a glossary

The format of the glossary depends on personal choice and the media you are using. The key to any format is that the terms should be easily recognisable. You can do this by highlighting the onwards (making them bold), arranging the terms in alphabetical order (a must), making the cross-references visible etc.

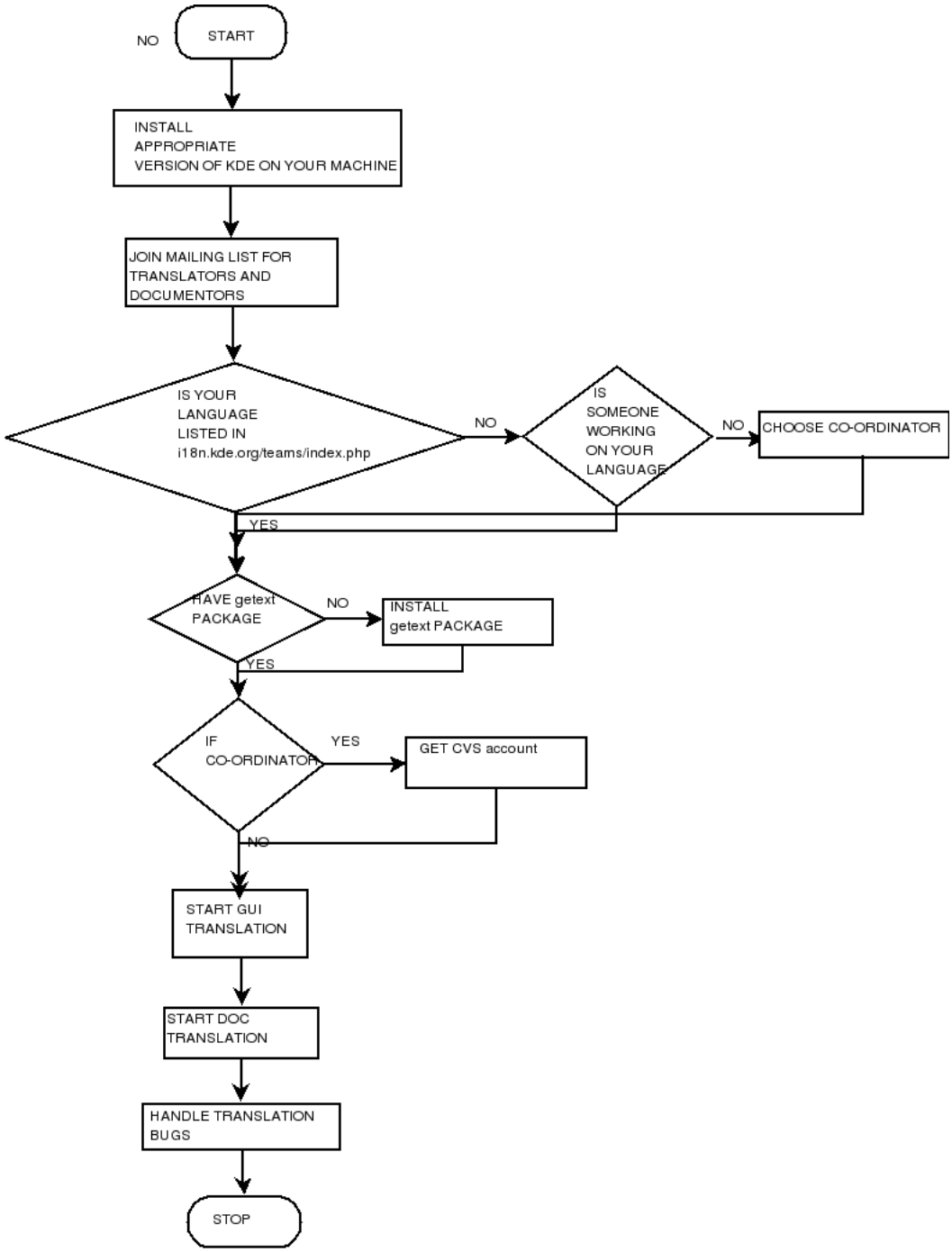
Chapter 12. KDE Localisation

12.1. Background

Read if: you are not aware of the KDE localisation process.

This chapter guides you on how to localise the KDE Desktop Environment. KDE is a free software Graphical desktop environment for Linux and Unix workstations. KDE intends to fill the need for a user-friendly desktop on Unix workstations, similar to the desktop environments found in MacOS and Microsoft Windows. A number of applications have been built for the K Desktop Environment: koffice, kmusic, kdraw, kdemedia are just to name a few.

Figure 12-1. KDE localisation - Roadmap



The above figure shows the KDE localisation process. The sections that follow essentially describe this entire process in detail.

Be sure of what version of KDE you are localising and install that version on your machine. Though you can do your translation work for KDE for an older version than the one the translation is meant for, it is recommended to have the target version at least for testing purposes. In order to localise the KDE Desktop environment, you need

to know about the following things:

- Things to be done in order to bring a new language like hypolan to KDE.
- GUI translation - Where to find the POT files, to-dos for GUI translation, checking and committing your translation work.
- Translation of documentation

We will discuss each of these topics in detail in this chapter.

12.2. Things to be done to bring hypolan to KDE

Read if: you want to know how-to start with KDE localisation process.

The first thing you have to do is join the mailing list for translators and documenters. This is where you will have to post your queries and this is the place which will give you information relating to any changes in the translation procedure. You can join the list either from the website (https://mail.kde.org/mailman/listinfo/kde-i18n-doc/) [KDE1] or by sending a mail to kde-i18n-doc-request@kde.org with subscribe as the subject. Next you have to check if hypolan figures in the list of languages and coordinators in KDE (http://i18n.kde.org/teams/index.php) [KDE2]. If it does then you have to simply write to the coordinator(s) asking him/her/them how to proceed. In case hypolan does not figure in the list then you can post a message to the mailing list asking if someone is already working on Hypolan or would like to join you. If no one on the list replies then you would get a message from the GUI translation coordinator probably suggesting you to become the coordinator for Hypolan and proceed as described in the following sub-sections. If you do get a response, based on the response, work with them. Working in a group ensures wider reach for your work, better feedback etc.

12.2.1. Available resources and CVS

- Make sure that the gettext package (Chapter 10) is installed on your machine and make yourself familiar with the package.
- Try to be familiar with translation tools like KBabel, KTranslator or (X)Emacs in PO mode. You can find a list of tools at Chapter 18.
- Be sure to get familiar with CVS (if you are the hypolan team coordinator) or with anonymous CVS or CVSUP (if you are just a translator). CVS is a version control system for ASCII files which maybe changed by multiple users across the network. In KDE "CVS" primarily refers to the "cvs.kde.org" server where all the source code for KDE programs, documentation, translation and everything related to KDE are maintained.

If you are the coordinator, you should get a CVS account for KDE. In order to get a CVS account you have to send a request to the KDE System administrator along with a user name and encrypted password. You can encrypt your password using the following command:

perl -e "print crypt('{passwd}','{xy}'),\"\n\";"

where {passwd} has to be replaced with your password and {xy} should be replaced with two random characters. Please remember that if your password contains a '$' sign then it should be escaped with a backslash. For example, if your password is abc$123 then it should be written as abc\$123.

As a rule you should have only one account for your team. But of course, you can have more if they are needed.

People having a CVS account have a local client, corresponding to this CVS server, set up on their machine. The CVS server mirrors the source tree on the remote machine. This local client allows them to merge their work back to the remote source tree system as and when required.

One thing you should remember is that when you are creating directories in CVS tree, the directories should contain at least the files Makefile.am and .cvsignore. These are needed for internal administration and proper compilation. You can copy these files from other language directories like kde-i18n/de/ but while doing so please don't forget to replace the abbreviation "de" in Makefile.am with your own "official" language abbreviation. In most cases, this abbreviation follows the respective ISO standards (http://www.loc.gov/standards/iso639-2/englangn.html) [TLC4].

12.2.2. Starting Actual translation

Start your translation with the GUI interface. It is recommended that you use the KBabel tool (Refer Section 19.4) to do the translation. You should first start translating the kdelibs.pot file because its contents are spread across various KDE applications as menus. This can be followed with desktop_kdelibs.pot and desktop_kde-i18n.pot. You can then proceed with applications in kdebase. How to go about doing these translations is explained in detail in the section on GUI.

The official rule to release any language version says that 90% of kdelibs.pot, 100% of desktop_kdelibs.pot and desktop_kde-i18n.pot and around 75% of kdebase have to be translated. But usually the team coordinators can decide when their language is "ready for release" and inform the concerned people.

12.3. GUI Translation

Read if: you are not aware of where to find the PO files for GUI translation of KDE

and how to go about the GUI translation.

As mentioned earlier, start KDE localisation with the translation of its GUI. This involves obtaining the necessary POT and POT files, translating them and then committing back the changes made. The following sub-sections explain these processes in detail.

12.3.1. POT and PO files

KDE has all the GUI completely internationalised and hence all you need to do is to translate all messages in its .po files. Don't know about POT/PO files ? See Section 10.1 for description on POT and PO files.

The PO files can be obtained from the corresponding POT files by just loading the POT file in any ASCII editor or any specialised programs like KTranslator, KBabel etc. and saving it as a .po file. You can now translate the strings inside the PO file into Hypolan. If you do not know about the structure of PO files or how and what to translate in PO files just check Section 10.1.

The POT files and all current translations of these can be found in a separate folder called kde-i18n and are organised into different sub-folders for different languages. [http://i18n.kde.org/translation-howto/gui-translation.html]

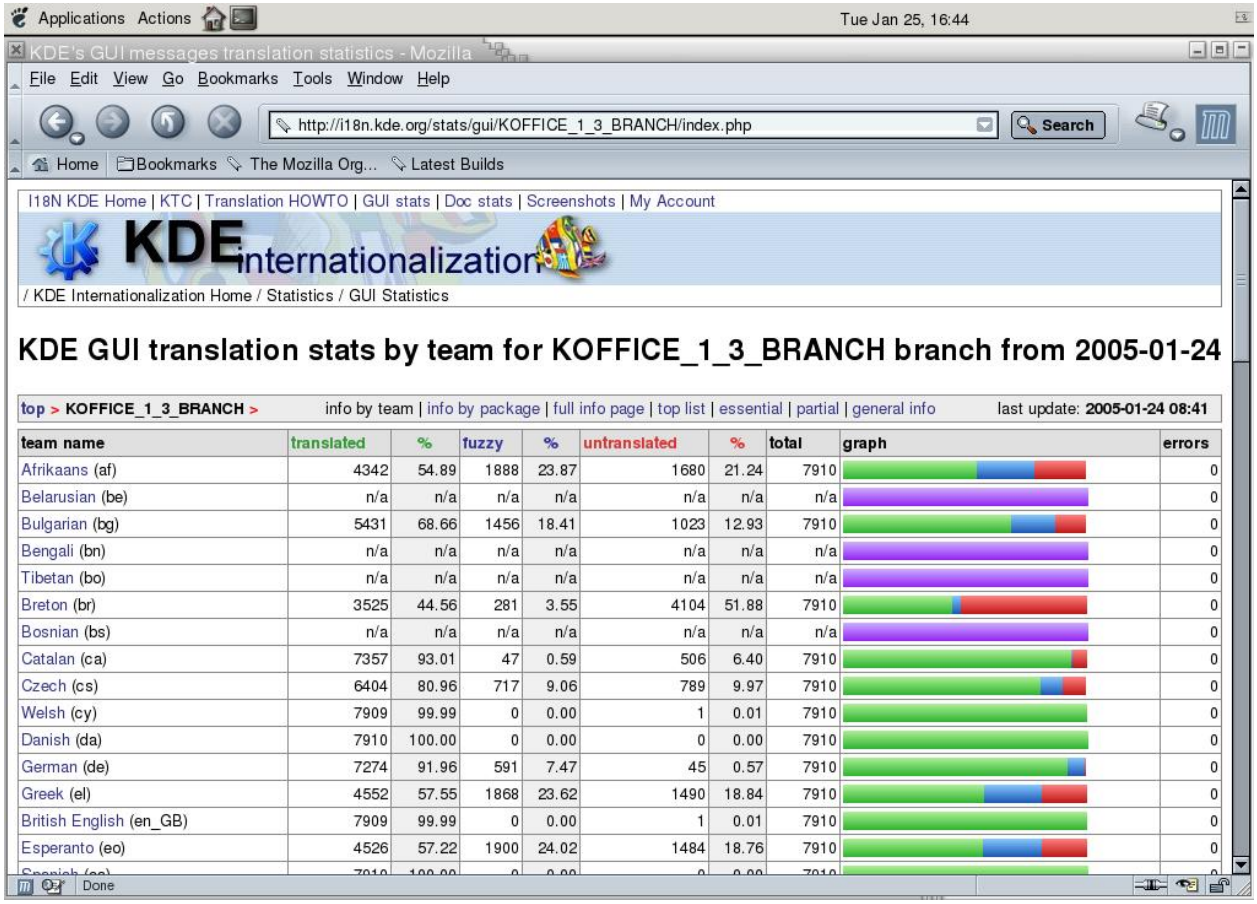
- The POT files are in the folder http://webcvs.kde.org/kde-i18n/templates/
- Language specific PO files for various applications are in the folder http://webcvs.kde.org/kde-i18n/$LANG/messages/{package-name}/. e.g.) http://webcvs.kde.org/kde-i18n/de/messages/kdebase/konqueror.po for a German file. Similarly if 'hl' is the 2-letter ISO code for Hypolan then the konqueror.po file for Hypolan would be found in http://webcvs.kde.org/kde-i18n/hl/messages/kdebase/konqueror.po
- language specific documents are in the folder http://webcvs.kde.org/kde-i18n/$LANG/docs/{package-name}/{appfolder/index.docbook. e.g. http://webcvs.kde.org/kde-i18n/hl/docs/kdeutils/kdiff/index.docbook

12.3.2. To-Do List for translating PO files

First check if translation work for Hypolan has already started. Try to figure out which programs have not been translated yet. There are two ways of doing this:

- You can find out from KDE Internationalisation status table (http://i18n.kde.org/stats/gui/) [KDE3]. The following figure shows the status of KOffice translations in various languages.

Figure 12-2. KOFFICE translation status as of 24-Jan-2005



In the above figure, green stands for 'translated messages', red for 'untranslated messages', blue for 'fuzzy messages' and violet for 'information not available'.

- Compare the POT files with the PO files under the hypolan directory. If there are no PO files for a particular program in http://webcvs.kde.org/kde-i18n/$LANG/messages/{package-name}/ then there is a pretty high probability that the program has not been translated. However it is better to contact the team co-ordinator (as mentioned earlier) to find out if someone is already working on the program. If no PO file has been created for the program then load the corresponding POT file into an ASCII editor and save it with a .po extension in the folder where it should be present.

See Chapter 11 for guidelines on translation.

Once you have the PO files ready you are required to fill the header information (who,when etc. as well as removing the first "fuzzy" tag) and fill the empty strings with translation. One thing that should be kept in mind while translating PO files is that if you are working on an ASCII editor don't remove the initial "msgid/msgstr". But make sure you remove the "fuzzy" which is initially there. Otherwise, you will get a parse error while compiling the PO file. For more details on this see KDE site

(http://i18n.kde.org/translation-howto/gui-todo-lists.html) [KDE4].

Since the KDE programs are constantly being improved, the GUI texts keep constantly changing. Hence already translated PO files will need to be updated frequently. In order to check which PO files need updation either CVS update your working directory or refer to KDE's GUI messages translation statistics (http://i18n.kde.org/stats/gui/) [KDE3].

12.3.3. Checking and Committing your work

The following things should be kept in mind before committing your work to the CVS source tree:

- Before committing your work please ensure that you have validated your .po files using the gettext package. The command for this is msgfmt. Don't know about the gettext package ? See Chapter 10
- PO files may contain comments on the exact meaning of a phrase. e.g.) The word "home" can refer to the home directory or a key in the keyboard. Be sure that such context information does not show in your translation.
- Arguments that are used for dynamic values must show up in the translation also. e.g.)msgid "Unknown package type: %1"

Tools like KBabel automatically ensure most of these for you. If you are not using any of such tools, then you must ensure these manually.

12.3.4. Peculiarities and Difficulties in GUI Translation

GUI translation is not as trivial as it may sound. It does have some peculiarities. The following points should be taken into account while doing GUI translation:

- You have to make sure that same letter is not used twice to mark an shortcut key within one menu. This can be done by performing semiautomatic tests for finding existing duplicates. There are such tests. You can refer Compilation, Context and Accelerator Checks (http://i18n.kde.org/translation-howto/check-gui.html#check-context) [KDE5].
- Some PO files contain context information to explain the context of the message string to the translator. For example there could be context information to specify if home refers to the home directory or to the keyboard key. Such information always begins with a "-:" and should not be translated.
- Special characters like quotes ("") must be escaped, that is, must be preceded with a back slash. Unescaped quotes would be interpreted by programs to be end of of msgstr.

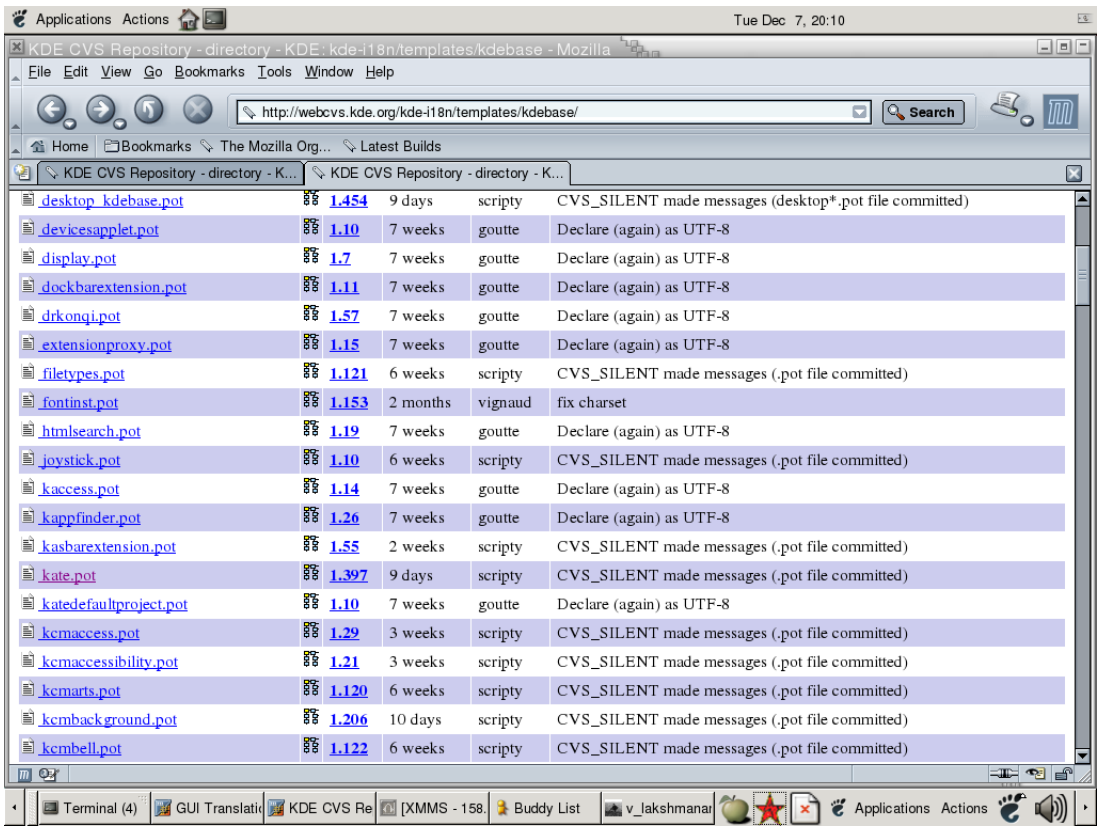
- One problem with languages whose average length and width of words are greater than others is that their translations are often out of proportion and cause the words to go outside the specified area in the menu/dialog box/etc. This however can only be tested after translation.
- Sometimes even if all the strings are translated, English text appears in some places during a run. There could be many reasons for this; one of them being that the particular text has not been made "i18n conform" by the author. If such a problem occur, report it to bugs.kde.org (http://bugs.kde.org/) [KDE6].

12.3.5. Case-study : KATE in Hindi

In this section we will explain the process of obtaining PO files for localising the application KATE. KATE is a multi document editor. KATE has been moved to the kdebase package and is a built in part of the KDE desktop environment.

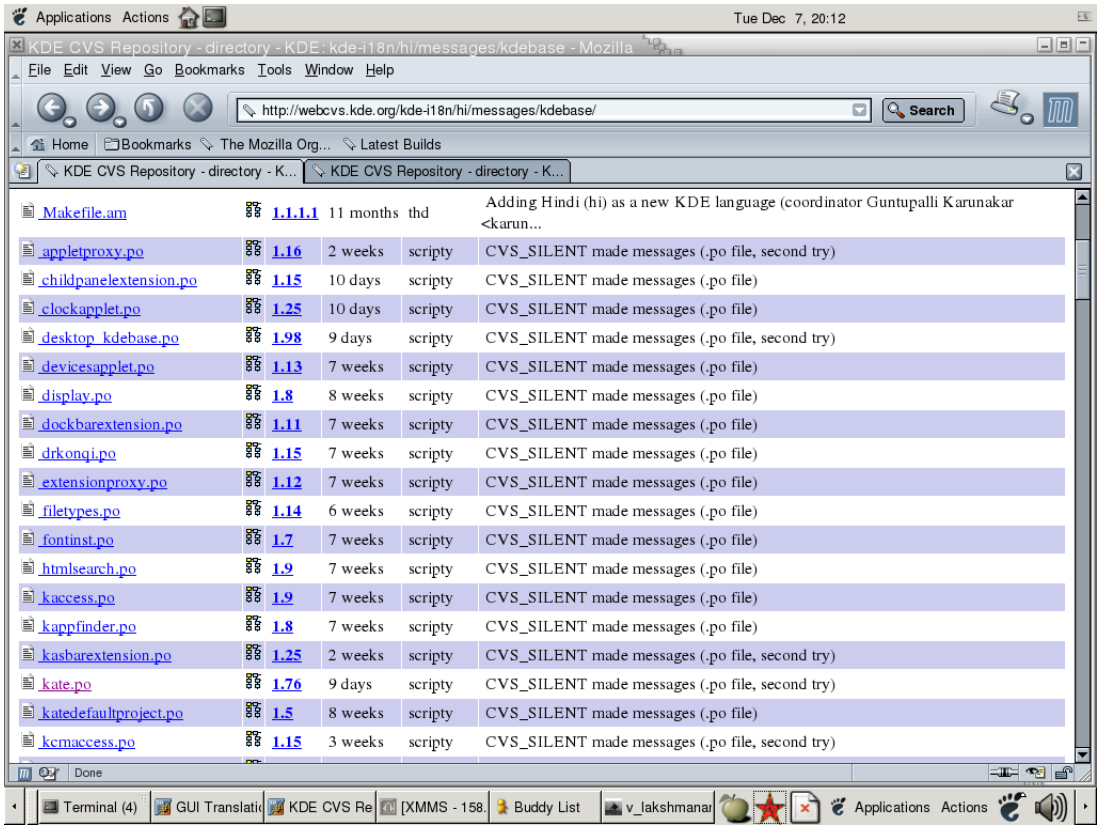
- The first step is to obtain the POT and PO files.

  The POT files can be found in the folder http://webcvs.kde.org/kde-i18n/templates/kdebase/. The following figure shows this directory structure and the "kate.pot" file.
  Figure 12-3. kate.pot file for Hindi



- The PO files for Hindi language can be found at here (http://webcvs.kde.org/kde-i18n/hi/messages/kdebase/) [KDE7]. The following figure shows this directory structure and the "kate.po" file in it:

Figure 12-4. kate.po file



- There are no language specific documents available for KATE as of today.

Having obtained the "po" files you can start your translation and proceed as explained in the previous sections.

12.4. DOC Translation

The documentation and on-line help for KDE programs is written in XML and the DTD used for this is Docbook. But for the convenience of the translators, the files to be translated are in Unicode format with .pot and .po extensions. The POT files and the translated po files and DocBook files for all languages are located in the KDE package named kde-i18n. For the package "ABC" you will find

- the POT files in the directory kde-i18n/templates/decs/(ABC)

- the translated PO files in the directory kde-i18n/$LANGUAGE/messages/docs/(ABC)/
- The English documentation files in the directory (ABC)/doc/ . You will need this file to generate the translated DocBook file
- the translated documentation (in DocBook format) in the directory kde-i18n/$LANGUAGE/docs/(ABC)

Most of the documentation translation process is similar to GUI translation. If there are screenshots in the documentation then you will have to create a localised version of the same. You can do it with the help of KSnapshot.

For a more detailed how-to on Documentation Translation please refer to [KDE8]

12.5. Handling translation bugs

Read if: you are not aware of the bugs handling procedure of KDE.

Translations are a part of the KDE bug tracking system now. Translation coordinators will be receiving the following:

- The output of automatic PO files testing from the Bug tracking system and not from the i18n coordinator
- Reports from users on what they think is bad or wrong about the translations.

The KDE Bug-tracking system is explained in depth at bugs.kde.org (http://bugs.kde.org/) [KDE6]. The following sections describe in brief the process of submitting a bug report and working on a bug report.

12.5.1. Submitting a Bug Report

A bug report needs to be sent to submit@bugs.kde.org . This report will be given a number and the sender will be acknowledged with the same. The number is also forwarded to the kde-bugs-dist. If the submitter also names the package, then the number will also be forwarded to the package maintainer. The subject of the above mentioned mail will have the bug number added as bug#nnn and the reply-to will have both the address of the submitter and "nnn@bugs.kde.org".

12.5.2. Closing Bug Reports

Translators who receive the bug report, should send a reply to the report after correcting it. The reply will goto the submitter and also "nnn-done@bugs.kde.org" or "nnn-close@bugs.kde.org" (instead of just nnn@bugs). The id of the submitter will automatically be included in the "to" field.

12.5.3. Followup Messages

In case the translator does not want to close the report after making the changes, he can do so by sending the mail without modifying the "to" filed. Thus, in this case the reply will be sent to the submitter and "nnn@bugs". The bug tracking system will file the reply and forward it to kde-bugs-dist. The bug will not be marked as closed in this case.

Do not use the "reply to all recipients" or "followup" feature of your mail program unless you intend to edit the recipients substantially. In particular, do not send the followup message to "nnn@bugs.kde.org" and to "submit@bugs.kde.org" because system will then get two copies of it and each will be forwarded to kde-bugs-dist.

Chapter 13. GNOME Localisation

This chapter guides you on how to localise the GNOME Desktop Environment (http://www.gnome.org) [GNM1].

13.1. Background

GNOME stands for GNU Network Object Model Environment. It is one of the mature desktop environments available for GNU/Linux. GNOME is a Free Software (http://www.gnu.org/philosophy/free-sw.html) [FOSS3] and part of the GNU (http://www.gnu.org) project [FOSS1].

The GNOME Desktop Environment consists of developer libraries and a set of applications. You will have to localise both the developer libraries and the software packages corresponding to the applications of your interest, to use GNOME in your local language.

The core libraries of GNOME are gtk+, glib, gconf, gail, libgnome, libgnomeui etc.

The main applications are gdm (login window), gedit (text editor), gnome-terminal (terminal emulator window), evolution (mail client), epiphany (browser), metacity (window manager), nautilus (file explorer) and gnome-desktop (menus on the desktop). It also includes various applications for audio/video, file viewers (including gpdf, gv etc), CD writing, various useful applets, desktop theme selector, printing tools etc. GNOME also includes office tools like gnumeric (spreadsheet), dia (diagram editor), planner etc.

The minimal things you need to translate for using GNOME in your language are gdm, gnome-desktop (for the menus on the desktop), Abiword (if you have no other word processor), gedit, gnome-terminal, evolution, epiphany (If you are not using any other browser), gpdf, gnome-panel, nautilus and libgnomeui (from developer-libs) etc.

Localisation of GNOME Desktop Environment involves the following steps :

Initiating the Translation Process,

Getting the files for the translation,

Translating the files,

Testing the translations,

Submitting the files back to the GNOME source tree,

13.2. Initiating the Translation Process

Once you have decided to localise GNOME desktop environment, you should find out if there is a team working on GNOME in your language. You can check this by going through the list of current language teams working on GNOME. This list is hosted (http://developer.gnome.org/projects/gtp/teams.html) [GNM2] on the GNOME translation project website (http://developer.gnome.org/projects/gtp/teams.html) [GNM3].

If the translation team already exists for your language, you can join the team by contacting the team coordinator and then participate in the translation process.

If a translation team does not exist for your language, you need to set up one. You can set up a team by writing a mail to the gnome-i18n@gnome.org mailing list. The mail should have a subject like "New team for [your language name] ([your language code])" and contain the name and the email of the person who will be the new coordinator for that language. The coordinator can be either you or anyone else from your team members.

Once your team is approved by the GNOME translation team, your language will be added in the language status page (http://l10n-status.gnome.org) [GNM4] on the GNOME Translation Project Website. The coordinator of the team is the primary contact person for the team, and responsible for organising the translations.

GNOME has its CVS repository (containing source code and translatable files of GNOME) kept online (http://cvs.gnome.org/viewcvs/) [GNM5]. Anyone can download the files from the CVS; but only a few authorised people can commit the changes back. On approval of the team, the coordinator of the team will get a CVS account so that he can update the translations. So he will be the person who commits (submits) all the translations for your language.

Normally, first-time translators for the GNOME project are encouraged to send their work for submission to existing GNOME CVS account holders, who are more than obliged to help. As a frequent GNOME translator, one can request her own CVS account.

The coordinator is required to be familiar with CVS. If you are the coordinator and you are not familiar with CVS, please go through Section 19.1.2. Also, to know how to use GNOME CVS as a Translator check this (http://developer.gnome.org/doc/tutorials/gnome-i18n/translator.html) [GNM6].

13.3. Getting the files for the translation

For Localisation, GNOME uses Gettext framework. If you are the coordinator for your team, you should be familiar with the Gettext framework; Chapter 10 if you are not. The files you need to translate are of Section 10.1 format. A PO file consists of a list of strings, extracted from the source of the original application, with space to put in your

translations. Each application under GNOME Desktop environment has separate PO files for each language.

There are various ways of getting these PO files.

The best way is to download the PO files from your language's status page (http://l10n-status.gnome.org/) [GNM4] on the GNOME Translation Project website (http://developer.gnome.org/projects/gtp/) [GNM3]. Please note that your language will have an entry in the status page, only if translations have already started in your language.

As you have already contacted the GNOME translation team, your language will have an entry in the status page. So you can download the PO files from the status page.

The status page shows the translation status of all the recent releases of GNOME. You need to select the release of interest to you, and then from the following page, select your language link. Within this, there will be separate folders for each application and for the developer libraries. The PO files are located within these.

Normally the status page for a particular language is located at...

`http://l10n-status.gnome.org/gnome-2.10/xy/index.html.`

In the above URL, "gnome-2.10" indicates 2-10 release of GNOME, and xy indicates your language code. Every language has an associated language code. For information on language codes, please see Section 6.2.

Your language status page contains the latest PO file for any given application, if the translations are already started for the application, else it will contain a POT file. A POT file is a PO Template file. The idea is, language teams can copy the POT file and produce a corresponding PO file by changing the extension to '.po'.

Alternatively, if you are working in a team as a translator, you can get the files from the team coordinator. Then you can complete the translations and give them back to the coordinator.

13.4. Translating the files

After all, translation is the major task you have to do for using GNOME in your local language. There are more than 25000 strings to be translated in the basic GNOME distribution covering evolution (mail client) and epiphany (browser).

In GNOME, there are two sections to be translated, namely, developer-libs and desktop. Common buttons and dialogue buttons come from the developer-libs. developer-libs contains around 3000 strings. It covers modules like gtk+, libgnome and libgnomeui etc. If you have less resources and time, you can think of only translating the strings which frequently show up on desktop.

Desktop part contains all the applications distributed with GNOME, described in Section 13.1 of this chapter. It is better to pick something very small at first to get used

to the translation process. This also helps your team in showcasing the output and getting more volunteers. We suggest applications like gpdf, gnome-desktop (menus) etc to start with.

Before starting translations, we suggest you to go through the Chapter 11.

As you are dealing with PO files, you should also be familiar with PO files. If you are not familiar with PO files please go through the guidelines on working with Section 10.1.

It is also important to create/use the glossary of important terms used in the translations. Using glossaries will save a lot of time and help maintain consistency in the translations among the translators, and among different applications. GNOME already has a glossary. If possible, translate this glossary before starting to translate the actual software and use the glossary while doing the translation. You can download the glossary from here (http://developer.gnome.org/projects/gtp/glossary/) [GNM7].

13.5. Testing the translations

Testing is very important in the process of localising any application. Before submitting your work back to the GNOME, make sure that the translations are usable and there are no bugs that makes the application crash during the build process.

To test, you will have to install those translations into your system as described below. Once installed you can test the translations in the context where they occur.

The machine needs MO file to link the translations. You can create this file by running the command ‘msgfmt -cv‘ on the ".po" file(See example below). The above command creates the "messages.mo" file in the current directory. This file should be placed in the /usr/share/locale/LL/LC_MESSAGES/ directory in your machine (LL is your locale code) and given the name of the application. For example, gpdf.mo, gedit.mo etc. Sometimes the application name requires a version number. In this case, you will have to look at the already existing file names for that applications in the other language directories. You will get these other language files in the /usr/share/locale directory. Please note that you need root privileges to put the file in the above directory.

```
msgfmt -cv gpdf.po // Creates messages.mo file in the current directory
mv messages.mo gpdf.mo
su  // Login as root
cp gpdf.mo /usr/share/locale/LL/LC_MESSAGES
```

After translating, and installing the translations, make them available to the public and give them instructions to install so that they can also test the applications. Please do rigorous testing to make sure that the applications are usable and stable.

13.6. Submitting the files back to the GNOME source tree

Once you have finished the translations and have tested them rigorously, you should submit the work back to the GNOME source tree. You can also submit partial translations (i.e., you need not wait till you have finished translating all strings) so that others can take and work on them. Only a person with CVS access can do these updates. So all the translations will be submitted by the team coordinator.

Please make sure that PO files are encoded in UTF-8, and there are no errors in the file format, before committing them to GNOME CVS. You can check this out by running 'msgfmt' command on the PO files.

```
msgfmt -cv xy.po
```

The above command gives the following output if the PO file is alright.

```
xy.po: UTF-8 Unicode PO (Gettext message catalogue) text
```

If the above command is not successful and throws an error saying the PO file is not encoded in UTF-8, then you can convert the PO file to UTF-8 using the command 'msgconv'.

```
msgconv -t UTF-8 xy.po > xy.po.new
mv xy.po.new xy.po
```

Now you can commit the translations with the command 'cvs commit', on the GNOME CVS server (http://cvs.gnome.org) [GNM8]. For detailed explanation on the use of CVS, please go through the Section 19.1.2.

13.7. Suggestions

- You can use tools like gtranslator (http://gtranslator.sourceforge.net/) [TLN3], poedit (http://poedit.sourceforge.net/) [TLN5], and Kbabel (http://i18n.kde.org/tools/kbabel/) [TLN4] for translating PO files.
- Don't forget to join (http://developer.gnome.org/projects/gtp/contact.html) [GNM9] gnome-i18n mailing list. It is required for coordinators as he is the interface for the developers community. You can join the mailing list by sending an email to gnome-i18n@gnome.org with the subject line "subscribe".
- Translating the documentation is as important as translating the actual software. As far as possible, translate the GNOME docs.
- If some other team is working on some other application in your language, be in touch with them and make sure that translations are consistent across the applications.

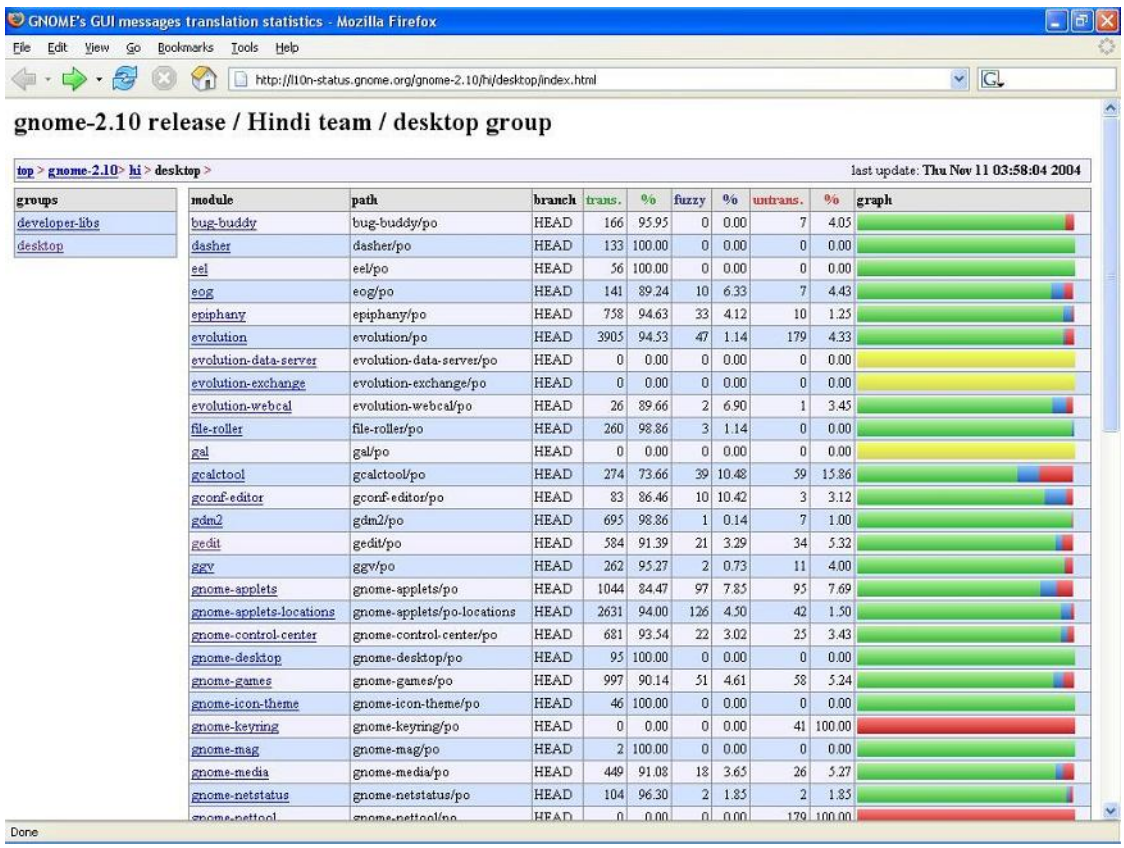
13.8. An Example: Localisation of gedit in Hindi

Above we have seen the general steps involved in localising any GNOME application. Now we will take one application (gedit) and look at the specific steps involved in localising it to Hindi language. Please go through the previous section of this chapter before reading further.

As you already know, the localisation of gedit (or another GNOME application) includes downloading the relevant PO file, translating it, testing the translations and submitting them(the PO file) back to the GNOME source tree, so that others can use/update your work. As you are localising in Hindi and for version GNOME-2.10, you can download the file from...

`http://l10n-status.gnome.org/gnome-2.10/hi/desktop/index.html`

The above page looks like this...
Figure 13-1. GNOME localisation: status page



After visiting the above page, right click on gedit (red in colour, in the above image) link and click on "save link as". It will save the gedit.hi.HEAD.po file in your machine.

Now translate the downloaded file using any of the available Translation Tools.

After you have finished translating the PO file, you should test your translations. The testing process can be done as explained above. You need to do the following...

```
cp gedit.hi.HEAD.po gedit.po
msgfmt -cv gedit.po // Creates messages.mo file in the current directory
mv messages.mo gedit.mo
su  // Login as root
cp gedit.mo /usr/share/locale/hi/LC_MESSAGES
```

Now you open gedit with the following command...

```
LANG=hi_IN gedit
```

If there are no bugs in your translations, gedit will open up in Hindi using the translations you have done. You can test the translations, modify the PO file accordingly and when everything is ok, submit it to GNOME.

Chapter 14. Mozilla Firefox Localisation

This chapter guides you on how to localise the Mozilla Firefox (http://www.mozilla.org/) [MOZ1] web browser.

Localisation of Firefox involves the following task.

Initiating the Translation Process,

Getting the files for the translation,

Translating the files,

Packaging the translations and testing them,

Submitting your work back to the community,

These are described below.

14.1. Initiating the Translation Process

Once you have decided to localise Mozilla firefox, you might want to check if a team is already working on Firefox in your language. This is important because, it will reduce your efforts as you don't have to start from scratch. You can check this by going through the list of current language teams working on Firefox. This list is hosted (http://www.mozilla.org/projects/firefox/l10n/) [MOZ2] on the Firefox localisation website. After visiting the above URL, scroll down a little and you will get the list of official/unofficial teams working on Firefox. A team is unofficial if it is not registered with Firefox but working on its localisation.

If the translation team already exists for your language, you can join the team by contacting the team coordinator and then participate in the translation process. You will get the coordinator's email address from the list of localisation teams hosted on the Firefox website (above URL).

If the translation team does not exist for your language, you need to set up one. You can set up a team by writing a mail to the Mozilla staff at mlp-staff@mozilla.org. The mail should contain the name and email address of the coordinator for your team. The coordinator of the team is the primary contact person for the team, and responsible for organising and maintaining the translations.

Next you should set up a team for localising the firefox browser. We suggest you to go through the general guidelines in Open Source Software.

Once your team is approved by the Mozilla translation team, your team will be added to the list of official Firefox teams. You can update your work more frequently if you are part of an official team.

Firefox has CVS server running. Anyone can download the files from the CVS server but only a few authorised people can commit the changes back. On the approval of the team, the coordinator of the team might get a CVS write access, so that he can update the translations. So he is the person who commits (submits) all the translations for your team.

A coordinator is required to be familiar with CVS. If you are the coordinator and you are not familiar with CVS, please go through CVS for guidelines on working with CVS.

14.2. Getting the files for the translation

You can translate firefox in two ways.

One is to get the files from the CVS tree of firefox, translate them and submit it back to the source. Your language support will be included in the next release of Firefox, i.e., your language will have its own language pack in the next release of Firefox.

Other way is to download the English language pack and translate it using the translate (http://translate.sourceforge.net/) [OOL7] tool. In the end you will create a language pack for your language and submit it back to the Firefox.

Either way is ok based on your interest.

If you translate using CVS, the translations will be available in the source tree of Firefox, so that anyone can download and reuse/enhance them.

If you translate using translate, you will make your own language pack and it can be kept on the Firefox site so that others can use it, translate it etc. This language pack will be available for a particular version of Firefox.

14.2.1. Getting the files from the source tree

The files to be localised are kept in two central locations in the Firefox source tree. The core localised files shared between Thunderbird(the email client) and Firefox are located (http://lxr.mozilla.org/aviarybranch/source/toolkit/locales/) [MOZ5] at mozilla/toolkit/locales in the CVS tree (http://lxr.mozilla.org/aviarybranch/source/) [MOZ7] The localised files which are used only by Firefox are located (http://lxr.mozilla.org/aviarybranch/source/browser/locales/) [MOZ6] at mozilla/browser/locales.

You will have to download both of these to see Firefox in your language. You can download these files in various ways.

The best method is to checkout the files from the Firefox CVS repository. You can do this by the following commands.

```
$ export $CVSROOT=:pserver:anonymous@cvs-mirror.mozilla.org:/cvsroot
$ cvs login  //Logging in to anonymous@cvs-mirror.mozilla.org
```

Enter the password : anonymous

```
$ mkdir firefox
$ mkdir firefox/browser
$ mkdir firefox/toolkit
$ cd firefox/browser
```

Please note that, Firefox 1.0 source is not on the main CVS "trunk" but rather from a branch, named AVIARY_1_0_20040515_BRANCH. So In order to check out the locale files from this branch, you should use the -r flag with CVS:

```
$ cvs checkout -r AVIARY_1_0_20040515_BRANCH mozilla/toolkit/locales
$ cd ../toolkit
$ cvs checkout -r AVIARY_1_0_20040515_BRANCH mozilla/browser/locales
```

The above commands will download the files to firefox directory in two sub directories. Note that, anyone can checkout (download) files from the CVS repository, no special rights are required.

If you are not familiar with CVS you can download the entire source of Firefox from any Mozilla mirrors and then extract the locale files. Or alternatively you can download a ZIP of the locale files, which will be provided at the time of localisation-freeze before a release. Downloaded zip files can be unzipped by using WinZip (http://www.winzip.com) [WZP1].

14.2.2. Getting the language pack

If translations are not started for your language, you will have to download the English language pack here (http://ftp.mozilla.org/pub/mozilla.org/firefox/releases/) [MOZ3]. It the translations are already started then you can download your own language pack from here (http://www.mozilla.org/projects/l10n/mlp_otherproj.html#ffox_contrib) [MOZ4].

14.3. Translating the files

The files you have to translate are DTD files. There are various ways of translating these files. If you are translating Firefox through CVS, i.e., downloaded the files from source tree of Firefox, you will have to translate the DTD and properties files as it is and manually package them or submit it to the source tree.

Alternatively if you downloaded the xpi(the language pack), you can use the translate tool to do everything. You can convert the DTD files to PO files, translate them and convert them back to the DTD files. You can even create the language pack.

14.3.1. Translating using PO format

If the translations are not yet started in your language you will have to download the English language pack and do the translations from scratch.

The process :

- Converting the dtd/properties files to PO format (command: moz2po),
- Translating the PO files by using any of the available translation tools,
- Converting the PO files back to dtd format and creating the language pack (command: po2moz).

Above steps can be done as follows...

```
moz2po -i en-US.xpi -o hi-IN-files
```

The above command converts all the .dtd and .properties files that are compressed in the en-US.xpi (American English language pack) file and creates an equivalent PO file for each one of them, placing in the correct location of the tree under hi-IN-files(for Indian-Hindi).

If the translations are already done in your language and language pack already exists, you don't have to translate the entire en-Us.xpi for the new version of Firefox. In this case you need both the old hi-IN.xpi and new en-US.xpi. And the command is as follows...

```
moz2po -i hi-IN.xpi -o hi-IN-files -t en-US.xpi
```

It will use en-US.xpi as the template.

After translating, give the command :

```
po2moz -i hi-IN-files -o hi-IN.xpi -t en-US.xpi
```

It takes the already translated PO files contained in the hi-IN-files directory, convert them to the original .dtd and .properties files with the help of the files contained in the en-US.xpi and creates a language pack called hi-IN.xpi.

14.3.2. Translating the .dtd/.properties files

As you are translating through CVS, you will have to translate the .dtd and .properties files. You can use Mozilla translator for this. But it does not work well with the Indic scripts. So we suggest you to do it manually. These files contains the English strings and the space for you to put in your language strings. You will have to translate all the files you downloaded from the CVS and submit it back to the source tree.

14.4. Packaging the translations and testing them

You have already packaged the translations in the above step, i.e., you created a language pack for your language you can install it on your system by clicking on the file. Once you have installed the language pack, you can see Firefox in your language. So you can test the translations, modify them as required and re-package them as explained above.

14.5. Submitting your work back to the community

Once you are sure that the translations are correct, you can submit it back to the Firefox community.

If you translated using CVS you have to commit them back to the CVS. For that you need CVS write access. There are guide lines put up on the Mozilla site on this. Or you can send your files to a person who has CVS write access.

If you packaged the translations, i.e., you made your own language pack you can mail it to mlp-staff@mozilla.org. They will host it on the localisation site of Firefox, so that others can download and use/test/translate it.

Chapter 15. OpenOffice Localisation

15.1. Background

Read if: you plan to localise the OpenOffice suite

OpenOffice (also known as OpenOffice.org or OOo for short) is a collection of software applications which is used to our day to day office work. It is very similar in its interface and behaviour to the Microsoft office suit. The most current version (which can be downloaded from its CVS branch (http://anoncvs.services.openoffice.org/) [OOL16]) of OpenOffice at the time when this document was written is version 1.1.x. This office suite is undergoing a major upgrade and version 2.x of this software will be released soon. OpenOffice software package basically contains five tools:-

Table 15-1. Major tools of OpenOffice.org

Tool	File Extension	Description
Writer	sxw	A word processor/text editor.
Calc	sxc	A Spreadsheet.
Draw	sxd	A paintbrush/image editing utility.
Impress	sxi	A presentation tool like Powerpoint from Microsoft.
Math	sxm	A mathematical formula editor.
Template	sxg	A template for other documents.

Most of these applications use XML based file formats for storing documents. The Math Component of Calc uses MathML.

The description of localisation of OpenOffice software is not complete without the description of internationalisation of the software. There are two internationalisation framework available with the OpenOffice.org, in the subdirectories ../i18n and ../i18npool. The first one is older and only supports western languages. The second one had to be created in order to support other languages. The whole development effort of the OOo team is now concentrated on ../i18npool. We will ignore the former framework in this document.

When we talk of localisation of Openoffice we generally refer to two ways:-

1. L10N-Framework: In this framework various tools to merge and extract strings and messages have been merged under a generic tool called localize.
2. L10N (Native Language support):

The former is the preferred way to localise openoffice, and does not require one to be technically very sound. We discuss both the models here, with more details on the former model.

15.2. The CVS Tree Structure

OpenOffice development makes use of the Concurrent Versions System (CVS). CVS is generic software environment for version control and management. CVS is described in CVS. CVS organises files in a tree structure. There is a root, branches sub-branches etc to any desired level of nesting. This branching and sub-branching can go up to any level. The different versions of the Application/Software or the parallel updated/fixes of the Application/Software reside on the different branches or sub-branches of the CVS tree. There are a set of commands to login to a CVS branch/tree, access files from the branch/tree, read, download the files from branch/tree,commit the update to the branch/tree. Almost all commands and sub-commands of CVS work recursively when the name of the directory is specified as an argument. A sample CVS structure is given below.

```
$HOME
        |
        +--mydir
        |   |
            +--CVS
            |      (internal CVS files)
            +--Makefile
            +--file1.c
            +--file2.c
            +--file3.c
            +--parser.c
            +--man
            |    |
            |    +--CVS
            |    |  (internal CVS files)
            |    +--testc.1
            |
            +--testing
                 |
                 +--CVS
                 |  (internal CVS files)
                 +--progm.t
                 +--test2.t
```

The root, also known as Head, is the place where the development of next major release is being carried out. The maintainence and bug-fixes of older versions are carried out on the branches from the root also known as the master-branches. Now each of these master-branches have sub-branches where the regular updates are being carried out in parallel. This scheme is implemented so as to refrain from accidently changing the master-branches. Only after a stable fix and QA testing, changes are merged with those at the master-branches.

In order to localise OOo, first you need to login to the CVS tree using login option with CVS. Logging in is required to modify the code and submit back to the site.

```
cvs -d:pserver:anoncvs@anoncvs.services.openoffice.org:/cvs login
```

Then you need to check out the code from the CVS tree. e.g.

```
cvs -d:pserver:anoncvs@anoncvs.services.openoffice.org:/cvs
                        co -r OpenOffice643 OpenOffice
```

You can now modify these files. After you have updated and built the code in your directory, to update the code on the CVS tree you invoke the update option of CVS. e.g.

```
cvs -d:pserver:anoncvs@anoncvs.services.openoffice.org:/cvs update
                       -r OpenOffice643 OpenOffice
```

For more information on CVS visit CVS and for OpenOffice.org CVS usage visit here (http://tools.openoffice.org/dev_docs/OOo_cws.html) [OOL17].

15.3. Internationalisation Of OpenOffice

Internationalisation module of OpenOffice.org ensures that the suite is adaptable to the requirements of different native languages, their local settings and customs, etc without source code modification or recompilation. The settings are specified by the user in the beginning, well before the program is used. Once the settings are in place, the user can freely use it naturally according to his locale.

As discussed in the Introduction section all efforts of Internationalisation in OOo are now concentrated on the new framework (i18npool). The solver tree path for the i18n pool is <openoffice source Home>/i18npool and the path of old i18n framework is <openoffice source Home>/i18n. There is wide support for the different geographic regions of the world.

Internationalisation in OpenOffice is dealt under the following categories:-

1. Locale Data
2. Character Classification
3. Calendar
4. Break Iterator
5. Collator
6. Transliteration

7. Index Entry
8. Search and Replace

These components are implemented in different C/C++ files located under various directories. To add a new service or new property to the existing components you can either.

1. Derive a new class from the existing classes.
2. Overload/Override different methods in the available classes.

15.3.1. LocaleData

Locale data for a language is defined in an XML file whose name is <language>_<country>.xml, in the directory <openoffice source Home>/i18npool/source/localedata/data. For example, for Hindi, it must be defined as hi_IN.xml. The <language> in the template is an ISO-639 two letter code for a language which can be obtained here (http://www.w3.org/WAI/ER/IG/ert/iso639.htm) [OOL4]. The <country> in the template is an ISO-3166 uppercase two letter code which can be obtained from here (http://www.iso.org/iso/en/prods-services/iso3166ma/02iso-3166-code-lists/list-en1.html) [OOL5]. After making suitable changes in the concerned file, an entry must be placed in the makefile.mk file which is in the same subdirectory, for the corresponding .cxx file and .obj file. A sample is given below.

```
# --- Files --------------------------------------------------------

# Interim files generated by the saxparser executable, for dependencies
MY_MISC_CXXFILES = \
    $(MISC)$/localedata_af_ZA.cxx  \
        $(MISC)$/localedata_ar_EG.cxx  \
    ------------------------------------------------------
    ------------------------------------------------------
    $(MISC)$/localedata_gu_IN.cxx  \
    $(MISC)$/localedata_he_IL.cxx  \
    $(MISC)$/localedata_hi_IN.cxx  \
    $(MISC)$/localedata_hu_HU.cxx  \
    $(MISC)$/localedata_id_ID.cxx  \
    $(MISC)$/localedata_is_IS.cxx  \
    ------------------------------------------------------
    ------------------------------------------------------
        $(SLO)$/localedata_gu_IN.obj \
        $(SLO)$/localedata_he_IL.obj \
        $(SLO)$/localedata_hi_IN.obj \
        $(SLO)$/localedata_hu_HU.obj \
```

```
        $(SLO)$/localedata_id_ID.obj \
        $(SLO)$/localedata_ja_JP.obj \
    ----------------------------------------------------
    ----------------------------------------------------
```

The locale must also be added to the aDllsTable structure in the file <openoffice source Home>/i18npool/source/localedata/localedata.cxx. A sample is given below.

```
static const struct {
        const char* pLocale;
        const char* pDLL;
        const char* lLocale;
} aDllsTable[] = {
        { "en_US",  lcl_DATA_EN, "en" },
        { "en_AU",  lcl_DATA_EN, NULL },
        ----------------------------------------------------------
        ----------------------------------------------------------
        { "hi_IN",  lcl_DATA_OTHERS, "hi" },
        { "kn_IN",  lcl_DATA_OTHERS, "kn" },
        ----------------------------------------------------------
        ----------------------------------------------------------
```

Finally the symbols are added to the linker map file using the instructions provided in the file <openoffice source Home>i18npool/source/localedata/data/linkermapfile-check.awk.

15.3.2. Character Classification

In the file <openoffice source Home>/i18npool/source/characterclassification/, the class cclass_unicode provides the features and API's to implement the character related features. Feature like case switching is implemented as toUpper()/toLower() functions in class cclass_unicode. Now if our language contains some specific features that is not generalised in this class we can derive our own language specific class which inherits this class. The parser related features are also implemented for this class.

15.3.3. Calendar

Located in <openoffice source Home>/i18npool/source/calendar/ the files and classes in this directory provide the ability to support a variety of calendaring systems as per the needs of the locale.

15.3.4. Break Iterator

These components deal with issues like cursor positioning and selection, line breaks, traversing between the lines. The implementation files are located in the subdirectory <openoffice source Home>/i18npool/source/breakiterator/.

15.3.5. Collator

These components deal with the sorting/lexical order of the languages. The implementation files are located under the directory <openoffice source Home>/i18npool/source/collator/.

15.3.6. Transliteration

These components deal with the transliteration issues for different languages. The implementation files are located under the directory <openoffice source Home>/i18npool/source/transliteration/. This framework can be used to provide a menu item to convert our texts from one script to another script. This form of facility is very useful for languages of CJK family, where input is done in one script and later on it is converted to another script. Transliteration is provided in three categories namely 1) Ignore 2) OneToOne and 3) Numeric.

15.3.7. Index entry

These components are used to generate index in pages.The implementation files are located under the directory <openoffice source Home>/i18npool/source/indexentry/.

15.3.8. Search and Replace

These components support the search and replace functionality. The component is designed in such a way that regular expressions can be used for the search purposes. The implementation files are located under the directory <openoffice source Home>/i18npool/source/search/.

If your language is almost identical in these respects to an existing language, you can replicate the lines corresponding to that language and make any changes, if required. If there are major changes, then you may need more time here; please refer to the documentation in detail before making changes.

15.4. Localisation under L10N System

If you are a hard-core programmer this will be a place of your liking! The localisation process under L10N makes extensive use of C++ files. Localisation in this framework is done using a set of tools. You extract the strings from the application, translate the strings and then merge the strings in the application.

The first step in localising OpenOffice.org under L10N framework is to obtain a milestone workspace from the solver tree,which has been described earlier under the heading The CVS Tree Structure. (See Section 15.2)

Once you have downloaded the source files, the following needs to be done:-

1. Add a new Language to the Resource System

2. Add a new language to the Build Environment
3. Add a new Language to the Localisation tools
4. Extract, Translate, Merge Strings and Messages to and from the Source Code

Now we describe each of these processes one by one illustrating the tasks with example and identifying relevant tools.

15.4.1. Add a new Language to the Resource System

Add a MS-LANGID for your language in the file <openoffice source Home>/tools/inc/lang.hxx. These LANGID's are divided into 2 parts as primary language id which is represented by the lower 10 bits and sub-language language id which is represented by the upper 6 bits of the LANGID. If your language is not listed in this file then you need to define an entry for your language. This can be defined as LANGUAGE_USER_<your language>. These values must conform to the MS-LANGID rule for the user space. One must be careful while defining a value, so that it does not conflict with existing values. A fragment of the file <openoffice source Home>/tools/inc/lang.hxx is given below:-

```
#define LANGUAGE_DONTKNOW           0x03FF
#define LANGUAGE_NONE               0x00FF
#define LANGUAGE_SYSTEM             0x0000
#define LANGUAGE_AFRIKAANS          0x0436
#define LANGUAGE_ALBANIAN           0x041C
#define LANGUAGE_ARABIC             0x0001 /*primary only, not a locale!*/
------------------------------------------------------------------
------------------------------------------------------------------
#define LANGUAGE_BENGALI            0x0445
------------------------------------------------------------------
------------------------------------------------------------------
#define LANGUAGE_GUJARATI           0x0447
#define LANGUAGE_HEBREW             0x040D
#define LANGUAGE_HINDI                      0x0439
------------------------------------------------------------------
------------------------------------------------------------------
#define LANGUAGE_URDU_PAKISTAN      0x0420  /* variations of a language */
#define LANGUAGE_URDU_INDIA         0x0820
------------------------------------------------------------------
------------------------------------------------------------------
```

These LANGID's are as defined by Microsoft and is listed on their website (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/intl/nls_238z.asp) [OOL3] .

Now check to see if there is a mapping between LANGID of your language to its ISO name in the file <openoffice source Home>/tools/source/intntl/isolang.cxx . A subset of RFC 3066 describes Language identifiers. Therefore the possibilities are ISO 639-1,

ISO 639-2 and ISO 3166. ISO 639-2 is used when no ISO 639-1 code exists. ISO 3166 codes are used as the country codes. This is done basically to separate the languages with the same ISO 639-1 or ISO 639-2 codes. The core libraries yet do not support ISO 639-2 codes, but this will be supported in the future. A sample entry in this file is given below:-

```
static IsoLangEntry const aImplIsoLangEntries[] =
{
    // Lang (MS-LCID-Codes)             ISO639-1  ISO3166
    { LANGUAGE_ENGLISH,                     "en", ""   },
    { LANGUAGE_ENGLISH_US,                  "en", "US" },
    { LANGUAGE_ENGLISH_UK,                  "en", "GB" },
    { LANGUAGE_ENGLISH_AUS,                 "en", "AU" },
    3C
    3C
    ----------------------------------------------------------------
    ----------------------------------------------------------------
    { LANGUAGE_HINDI,           "hi", "IN" },
    ----------------------------------------------------------------
    ----------------------------------------------------------------
    { LANGUAGE_NEPALI,          "ne", "NP" },
    { LANGUAGE_NEPALI_INDIA,    "ne", "IN" },
    ----------------------------------------------------------------
    ----------------------------------------------------------------
```

For more information please see the file <openoffice source Home>/tools/inc/lang.hxx.

Now for the locales to be correctly shown in the listboxes, a UI visible entry must be added to the string list arrays of available locales in the <openoffice source Home>/svx/source/dialog/langtab.src.

In the file <openoffice source Home>/tools/source/rc/resmgr.cxx, add your Language to the switch case of ResMgr::GetLang(). Add your language to the static array of ResMgr::SearchCreateresMg(). Most of the languages have been added in these functions. But in case they don't exist there, add one, following the convention used for other languages.

```
const char* ResMgr::GetLang( LanguageType& nType, USHORT nPrio )
{
            ----------------------------------------------------------------
            ----------------------------------------------------------------
        switch ( nType )
        {
            ----------------------------------------------------------------
            ----------------------------------------------------------------
            case LANGUAGE_KOREAN:
            case LANGUAGE_KOREAN_JOHAB:
                return "82";
```

```
            case LANGUAGE_THAI:
                return "66";
            case LANGUAGE_HINDI:
                return "91";

            case LANGUAGE_ARABIC:
            case LANGUAGE_ARABIC_IRAQ:
            case LANGUAGE_ARABIC_EGYPT:
            ------------------------------------------------------------------
            ------------------------------------------------------------------
           case LANGUAGE_ARABIC_QATAR:
                return "96";
            ------------------------------------------------------------------
            ------------------------------------------------------------------
            default:
                return "99";
        }
}
            ------------------------------------------------------------------
            ------------------------------------------------------------------
}

ResMgr* ResMgr::SearchCreateResMgr(const sal_Char* pPrefixName,
                                                    LanguageType& nType )
{

 ------------------------------------------------------------------
 ------------------------------------------------------------------
 static const LanguageType aLanguages[] =
    {
        LANGUAGE_ENGLISH_US,
        ------------------------------------------------------------------
        ------------------------------------------------------------------
        LANGUAGE_KOREAN,
        LANGUAGE_KOREAN_JOHAB,
        LANGUAGE_THAI,
        LANGUAGE_HINDI,
        LANGUAGE_HEBREW,
        LANGUAGE_NORTHERNSOTHO,
        LANGUAGE_AFRIKAANS,
        LANGUAGE_ZULU
    };
ResMgr* ResMgr::SearchCreateResMgr(const sal_Char* pPrefixName,
                                                    LanguageType& nType)
{

            ------------------------------------------------------------------
            ------------------------------------------------------------------
```

```
static const LanguageType aLanguages[] =
   {
        LANGUAGE_ENGLISH_US,
        LANGUAGE_GERMAN,
        LANGUAGE_FRENCH,
            ------------------------------------------------------------------
            ------------------------------------------------------------------
        LANGUAGE_HINDI,
            ------------------------------------------------------------------
            ------------------------------------------------------------------
            ------------------------------------------------------------------
```

The new language which is to be added to the resource system must be supported by the resource compiler and hence must also be listed in the file <openoffice source Home>/rsc/inc/rsclang.c. The samples in this file is shown below.

```
LT( AFRIKAANS );
LT( ALBANIAN );
-----------------------------------------
-----------------------------------------
LT( ASSAMESE );
-----------------------------------------
-----------------------------------------
LT( ENGLISH_TRINIDAD );
LT( ENGLISH_UK );
LT( ENGLISH_US );
-----------------------------------------
-----------------------------------------
LT( GUJARATI );
LT( HEBREW );
LT( HINDI );
-----------------------------------------
-----------------------------------------
LT( URDU_INDIA );
LT( URDU_PAKISTAN );
-----------------------------------------
-----------------------------------------
```

15.4.2. Add a New Language to the Build Environment

The build environment uses the makefile, so we must add our new language and encoding that this language will use in the files <openoffice source Home>/solenv/inc/lang.mk and <openoffice source Home>/solenv/inc/postset.mk. The samples are given below:-

```
//This is a sample for the file lang.mk
```

```
------------------------------------------------
------------------------------------------------
greek$(LANG_GUI)=-CHARSET_microsoft-cp1253
turk$(LANG_GUI)=-CHARSET_microsoft-cp1254
korean$(LANG_GUI)=-CHARSET_UTF8
thai$(LANG_GUI)=-CHARSET_UTF8
hindi$(LANG_GUI)=-CHARSET_UTF8
extern$(LANG_GUI)=-CHARSET_UTF8
nsotho$(LANG_GUI)=-CHARSET_UTF8
afrik$(LANG_GUI)=-CHARSET_UTF8
zulu$(LANG_GUI)=-CHARSET_UTF8
------------------------------------------------------
------------------------------------------------------
----------------------------------------------------------------


----------------------------------------------------------------
//This is the sample for the file postset.mk

------------------------------------------------------
------------------------------------------------------
.IF "$(RES_HINDI)"!="" || "$(give_me_all_languages)"!=""
alllangext+=91
.ENDIF          # "$(RES_HINDI)"!="" || "$(give_me_all_languages)"!=""
completelangext+=91
hindi$(LANG_GUI)*=$(default$(LANG_GUI))
lang_91=hindi
longlang_91=hindi
langext_91=91
rsclang_91=-lgHINDI $(UTF8)
rescharset_91=$(hindi$(LANG_GUI))
RCLANGFLAGS_91+= -d HINDI
iso_91=hi-IN
------------------------------------------------------------
------------------------------------------------------------
```

If openoffice code for your language is not there,you can find it here (http://tmp.janik.cz/OpenOffice.org/Languages/OpenOffice.org-Languages.html) [OOL18]. If your language is not listed in this url, then you need to make a request at the l10n mailing list of openoffice for the code for your language. The codes in the above example for the language Hindi comes from this URL. This procedure will eventually be replaced by using ISO code in openoffice version 2.0

15.4.3. Add New Language to the Localisation Tools

We need to define our language in the file <openoffice source Home>/transex3/inc/export.hxx. Then the language is added in the

LANGUAGE_ALLOWED macro in the file <openoffice source Home>/transex3/inc/export.hxx. A sample is given in the below table.

```
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
#define HINDI                                   91
#define HINDI_ISO               "hi-IN"
#define HINDI_INDEX             30 /* This is the 30th Index entry
                                                                    in the fi

-------------------------------------------------------------------------------
-------------------------------------------------------------------------------

#define LANGUAGE_ALLOWED( n )  (( n != 0xFFFF )
    && ( Export::LanguageAllowed( Export::LangId[ n ] )) &&          \
 (( Export::LangId[ n ] == 01 ) || ( Export::LangId[ n ] == 03 )   \
 || ( Export::LangId[ n ] == 07 ) || ( Export::LangId[ n ] == 26 ) \
 || ( Export::LangId[ n ] == 27 ) || ( Export::LangId[ n ] == 28 ) \
 || ( Export::LangId[ n ] == 30 ) || ( Export::LangId[ n ] == 31 ) \
 || ( Export::LangId[ n ] == 33 ) || ( Export::LangId[ n ] == 34 ) \
 || ( Export::LangId[ n ] == 35 ) || ( Export::LangId[ n ] == 37 ) \
 || ( Export::LangId[ n ] == 36 ) || ( Export::LangId[ n ] == 79 ) \
         || ( Export::LangId[ n ] == 39 ) || ( Export::LangId[ n ] == 45 ) \
 || ( Export::LangId[ n ] == 46 ) || ( Export::LangId[ n ] == 48 ) \
 || ( Export::LangId[ n ] == 49 ) || ( Export::LangId[ n ] == 50 ) \
         || ( Export::LangId[ n ] == 53 ) || ( Export::LangId[ n ] == 55 ) \
 || ( Export::LangId[ n ] == 47 ) || ( Export::LangId[ n ] == 81 ) \
 || ( Export::LangId[ n ] == 82 ) || ( Export::LangId[ n ] == 86 ) \
         || ( Export::LangId[ n ] == 88 ) || ( Export::LangId[ n ] == 90 ) \
 || ( Export::LangId[ n ] == 96 ) || ( Export::LangId[ n ] == 42 ) \
 || ( Export::LangId[ n ] == 43 ) || ( Export::LangId[ n ] == 97 ) \
 || ( Export::LangId[ n ] == 66 ) || ( Export::LangId[ n ] == 91 ) \
 || ( Export::LangId[ n ] == 00 ) || ( Export::LangId[ n ] == 99 ) \
 || ( Export::LangId[ n ] == 77 ) || ( Export::LangId[ n ] == 36 )))
```

Then in the file <openoffice source Home>/transex3/source/export2.cxx, the new language must be added to the arrays Export::LangId, Export::GetLangByIsoLang, Export::GetIsoLangByIndex and it's symbolic representation must be added to the array Export::LangName. The samples are given below:-

```
USHORT Export::LangId[ LANGUAGES ] =
{
        // translation table: Index <=> LangId
        COMMENT,
        ENGLISH_US,
        PORTUGUESE,
       ------------------------------------------------------------------
```

```
        ------------------------------------------------------------------
         THAI,
         HINDI,
         ESTONIAN,
         SLOVENIAN
        ------------------------------------------------------------------
        ------------------------------------------------------------------
}


USHORT Export::GetLangByIsoLang( const ByteString &rIsoLang )
{
         ByteString sLang( rIsoLang );

         sLang.ToUpperAscii();

         if ( sLang == ByteString( COMMENT_ISO ).ToUpperAscii())
                 return COMMENT;
         else if ( sLang == ByteString( ENGLISH_US_ISO ).ToUpperAscii())
                 return ENGLISH_US;
        ------------------------------------------------------------------
        ------------------------------------------------------------------
         else if ( sLang == ByteString( THAI_ISO ).ToUpperAscii())
                 return THAI;
         else if ( sLang == ByteString( HINDI_ISO ).ToUpperAscii())
                 return HINDI;
         else if ( sLang == ByteString( ESTONIAN_ISO ).ToUpperAscii())
                 return ESTONIAN;
         else if ( sLang == ByteString( sIsoCode99 ).ToUpperAscii())
                 return EXTERN;

return 0xFFFF;
}

ByteString Export::GetIsoLangByIndex( USHORT nIndex )
{
         switch ( nIndex ) {
                 case COMMENT_INDEX: return COMMENT_ISO;
                 case ENGLISH_US_INDEX: return ENGLISH_US_ISO;
        ------------------------------------------------------------------
        ------------------------------------------------------------------
                 case HINDI_INDEX: return HINDI_ISO;
        ------------------------------------------------------------------
        ------------------------------------------------------------------
          case ZULU_INDEX: return ZULU_ISO;
          case EXTERN_INDEX: return sIsoCode99;
         }
         return "";
```

```
}

const ByteString Export::LangName[ LANGUAGES ] =
{
        "language_user1",
        "english_us",
        --------------------------------
        --------------------------------
        "thai",
        "hindi",
        --------------------------------
        --------------------------------
    "norwegian_nynorsk",
        "extern"
};
```

In the file <openoffice source Home>/tools/source/generic/l2txtenc.cxx define the new language and the encoding for the language in the method Langcode2TextEncoding(). A sample is given below in the table. The code to be used is-

```
#include "solar.h"
#include "l2txtenc.hxx"
#define COMMENT                                 0
#define ENGLISH_US                              1
-------------------------------------------------------------------------
-------------------------------------------------------------------------
#define THAI                                    66
#define HINDI                                   91
#define EXTERN                                  99

rtl_TextEncoding Langcode2TextEncoding( USHORT nLang )
{
        switch ( nLang ) {
                case COMMENT: return RTL_TEXTENCODING_MS_1252;
                case ENGLISH_US: return RTL_TEXTENCODING_MS_1252;
        ---------------------------------------------------------------
        ---------------------------------------------------------------
                case THAI: return RTL_TEXTENCODING_UTF8;
                case HINDI: return RTL_TEXTENCODING_UTF8;
                case EXTERN: return RTL_TEXTENCODING_UTF8;
                case NORTHERNSOTHO: return RTL_TEXTENCODING_UTF8;
                case AFRIKAANS: return RTL_TEXTENCODING_UTF8;
                case ZULU: return RTL_TEXTENCODING_UTF8;
        }
        return RTL_TEXTENCODING_MS_1252;
}
```

Then in the file <openoffice source Home>transex3/source/merge.cxx add the new language.The sample is given below.

```
USHORT MergeDataFile::GetLangIndex( USHORT nId )
{
        switch ( nId ) {
                case COMMENT: return COMMENT_INDEX;
                case ENGLISH_US: return ENGLISH_US_INDEX;
          ------------------------------------------------------
          ------------------------------------------------------
                case THAI: return THAI_INDEX;
                case HINDI: return HINDI_INDEX;
          ------------------------------------------------------
          ------------------------------------------------------
         case ZULU: return ZULU_INDEX;
         case EXTERN: return EXTERN_INDEX;
        }
        return 0xFFFF;
}
```

15.4.4. Extract,Translate,Merge Strings and Messages to and from the Source Code

The Localisation with L10N uses a collection of tools to extract and merge the strings for various file types. The various file types have various file formats and hence there are different tools to extract and merge strings and messages from them. A list of such tools is given below:

Table 15-2. OpenOffice localisation tools for various file types

Extension	Tools	Example
src,hrc	transex3	<openoffice s
lng	lngex	<openoffice s
xcd	cfgex	

The command line for these tools include options for both merging and extraction of the strings and messages. The extraction switches are [p,r,i,o]. The switch for merging the translated strings and messages are [m]. For example we invoke the transex3 in the following manner.

```
transex3 -p <project_name> -i <input_file> -o <output_file>
```

When run in extract mode each of these tools produce a tab separated output file which has '.out' as extension. You can see a sample of the file here (http://l10n.openoffice.org/textattr.src.out.txt) [OOL19].The table below describes the

format of this file. In actual file, each of these fields are tab separated.

Table 15-3. Fields of SDF/GSI files

Project	Source	Dummy	ResourceType	GroupID	LocalId	HelpId	Platform	Width	Langu

After the strings are extracted from the application these need to be translated. All the encodings(explained above in sections Add a new language to the Build Environment and Add a new Language to the Localisation tools) rules must be taken care of while translation is being done. When translation is over, these tools are used in the merge format to put the translated strings in the appropriate source files.

When run in merge format these tools require an inputfile format similar to the above output file format. The only difference being that each of these lines will have a corresponding translation on separate lines below in the desired languages.

15.5. Localisation under L10N Framework

The L10N framework is the easiest way to localise OpenOffice. One need not be technically very sound in a programming language like C/C++ to localise using this framework. You extract the strings from the application, translate the strings and then merge the strings in the application.

The first step in localising OpenOffice.org under L10N framework is to obtain a milestone workspace from the solver tree, which has been described earlier under the heading 'The CVS Tree Structure'. (See section Section 15.2)

After the workspace has been obtained, the strings need to be extracted out of the source code of the application to be translated. The earlier methods used various tools to extract different file formats, but in the L10N Framework, the functionality of these tools have been merged under one tool named 'localize'. The use of the tool is as follows:-

```
localize -e -i ISO-Code -l Languages -f OutputFile
```

Where,
ISO-Code: the ISO-code of the target language which has been described earlier.
Languages: The languages for which the strings have to be extracted.
OutputFile: A name for the tab separated file which contains the strings and messages, which are to be translated.

OutputFile is a tab separated file, which has been encoded in UTF-8 for all languages; it is commonly known as SDF/GSI file. This file stores the original strings and messages of the application. It contains the context(location) information of the message in the CVS folders and the strings themselves. Now all these strings

need to be translated in our desired language. A sample of this file format is described here (http://l10n.openoffice.org/L10N_Framework/Examples.inv/Language_49_as_source_and_fallback
The file types from which the strings are extracted can be one of the following types.
The types are denoted by the extension used for naming the files.

1. src,hrc (resource definition files)
2. lng (Strings and messages that are used in the setup program).
3. xrb (xml properties files)
4. xcd (xml configuration files)

Now we will describe the format of the SDF/GSI file. Given below are the fields of the file as columns of a table; in actual file columns are separated by a tab.

Table 15-4. Format of SDF/GSI file

project	source	dummy	resource	groupid	localid	helpid	platform	width	language	text id	help text	quick help text	title	time stamp

The text used for the translation is not real. This is just an example to explain the format. A quick summary of each column is given here (http://l10n.openoffice.org/L10N_Framework/Intermediate_file_format.html) [OOL20].

After a considerable amount of strings have been translated, we can merge the strings back to the application in order to release a version or to enable others to test the work so far. We can again use the localize command in the following way to do this.

```
localize -m -i ISO-Code -l Languages -f InputFile
```

where,

InputFile: It is the outputfile of the extraction step of localize tool. The only difference being that this file now contains all the translated strings as produced by the translation stage. The strings which are not translated are copies of the original strings.

After the extraction and merging of the strings are over, the code needs to be rebuilt again. The rebuild step (described later) can be found here (http://l10n.openoffice.org/L10N_Framework/How_to_localize_and_build_OpenOffice.html) [OOL21].

15.6. Tools useful for Translation

To ease the process of translating to and from openoffice GSI/SDF files, we can use the tools which comes with the translate package from here

(http://translate.sourceforge.net/) [OOL7]. The two tools which are commonly used for translation purposes are listed below.

1. oo2po: This tool converts the OpenOffice.org specific GSI/SDF files to PO files.
2. po2oo: This tool converts PO files back to the OpenOffice.org specific GSI/SDF files.

15.6.1. How to install and use translate tools

1. Download the tar file for the latest version of the package (e.g translate-0.8rc1.tar.gz), and run the command,

   ```
   tar -xzf translate-0.8rc1.tar.gz
   ```

   This will create a new directory which contains all the tools of translate package.

2. Run the following command in the newly created directory.

   ```
   ./setup.py install --home=/tmp/SOMEWHERE
   ```

3. Set the path of the python files for the environment variable PYTHONPATH.
4. Now you can use tools.

   ```
   usage: po2oo [--version] [-h|--help] [--progress PROGRESS] \
          [-i|--input] INPUT [-x|--exclude EXCLUDE] [-o|--output] OUTPUT \
          [-t|--template TEMPLATE] [--psyco PSYCO]
   ```

15.7. The build process For OpenOffice.org

15.7.1. Software requirements

The following softwares are required in order to configure and build OpenOffice.org

1. glibc
2. gcc
3. java jdk1.3.1 or above
4. csh
5. zip and unzip
6. gpc

7. GTK
8. Ant
9. Perl

15.7.2. Preparing for build process

1. In the directory <openoffice source Home>/config_office/, run the command

   ```
   ./configure
   ```

   This commands checks the software and hardware requirements necessary for the build process.

2. In the directory <openoffice source Home>/, run

   ```
   tcsh
   ```

3. Then, run

   ```
   source LinuxIntelEnv.set
   ```

   This sets all the environment variables necessary to run the build process afterwards.

4. run

   ```
   rehash
   ```

5. Then, run

   ```
   ./bootstrap
   ```

   This will create the tools, which are used while building the suite.

15.7.3. Building the Suite

To build the entire suite, run the following command.

```
dmake
```

To build the individual modules run the following commands in the subdirectory of the respective module

```
build
```

15.8. Locale information

In OpenOffice.org suite all the data specific to your locale (cultural data for your language/region), must be kept in XML files in UTF-8 format. In the OpenOffice.org Standard Development Kit (sdk) these files are contained under the directory <openoffice source Home>/i18npool/source/localedata/data/.The format of this file is given below.

```
<language name>_<COUNTRY NAME>.xml
```

For example for hindi the name of the file will be:

```
hi_IN.xml
```

To create a specific locale file for OpenOffice.org, copy a file which is most similar to your language, rename it in the form <iso name for your language>_<COUNTRY NAME).xml and make the appropriate changes to reflect the locale information as explained in Section 15.3.1.

Chapter 16. X Localisation

16.1. Introduction

By X localisation we mean localising the applications that directly interact with the X without going through a toolkit like GTK or QT. An example of such an application is Xterm. Though the localisation process is application specific, it is necessary to internationalise the X, for it to support such localisation. Since the X applications directly interact with the X, the inputs to these applications and the display of outputs of these applications is handled by the X. Hence it is not enough to localise the X applications alone but we should also prepare the X to handle inputs and outputs of other languages. This is what we mean by 'internationalising the X'. In this chapter, we briefly describe the approach to internationalise the X as done in X version 11 Release 6 and illustrate this in the context of Indix which is a project on enabling Indic scripts in GNU/Linux by making modifications at the X level.

16.2. Internationalisation in X11R6

Internationalisation of X is achieved by internationalising Xlib, which is a library of functions for displaying and texts and graphics and receiving inputs through possibly remote servers attached to displays and input devices. This library is a part of the X Windowing system. In X version 11 Release 6 (X11R6), internationalisation is achieved via X locales which is composed of the following components:

- XLC- This is the locale object. This provides information dependent on the language environment, for example , details about the character set.
- XIM- This takes care of text inputting. For more information on this, see Section 9.4
- XOM- This manages the text drawing, i.e. handles the outputs in X.

16.3. Indix

The current version of XOM, which handles the output in X, cannot support the display of complex scripts like the Indic scripts. For the display of these scripts additional libraries need to be written. The Indic project did changes at the XOM level, i.e., we could say that they came up with some kind of replacement for the XOM.

The project IndiX, with a view to enable Indic scripts on GNU/Linux has a different approach for i18n. IndiX achieves multilingual i18n by converting the rendering interfaces to support Unicode. IndiX has carried out these changes in X Windows framework by implementing the Indic shaping on the server side.

In IndiX, to display a multilingual string, client application invokes the rendering interface, implemented by the XDrawString() routine of Xlib. The XDrawString() routine is not modified and it packages the given string into the PolyText protocol request, and forwards it to the X11 Server. In the X11 Server, the string is separated into Indic and non-Indic runs based on the Unicode range allotted to the Indic script, say Devanagari. The Indic sequence is broken into syllables. For each syllable, the characters are reordered to facilitate proper display of the forms of those characters. After these transformations, the text is shaped. Using the CharMap in the font, the character codes are converted to glyph indices. Using substitution rules, groups of glyph indices are converted to indices of their ligatures or alternate forms. Then using information from the font, the final glyphs are positioned. The text shaping of Indic scripts is what makes it stand out from other scripts. The processing for non-Indic text run like Latin is comparatively trivial. The CharMap in font is sufficient to map the characters in string to corresponding glyphs. The entire pipeline is shown in the following figure.

Most display systems have efficient glyph rasterizing machinery that can take a sequence of glyphs, scale the glyph outline according to the point size and scaling of the display window, grid fit the points on the outline and scan convert the outline to a bit map. The rasterizing machinery generally has caching mechanisms so that frequently appearing outline will be rasterized only once. What comes out of the rasterizer is a pixmap that can be displayed by most devices in hardware.

Chapter 17. Kernel localisation

17.1. What is Kernel Localisation?

By Kernel Localisation we mean displaying the messages printed by the kernel in local languages. Note that most localisation efforts do not cover Kernel Localisation.

17.2. Approaches to Kernel localisation

There are a number of approaches suggested by different people to carry out Kernel localisation. One of the suggested approach is to have printk() strings available in all the possible languages and select the respective language string at run-time. Obviously, this approach would increase the size of the kernel by leaps and hence is often rejected.

Another approach suggested by Riley Williams is to add a unique message number to each message printed by the Kernel. Since the strings passed to the printk() method are already formatted and begin with a string like "<2>" which gives the log level of the message, Riley says why not have something like "<2.12345>" which could be used to index into a file of localised messages. Again, this approach is tagged unmaintainable.

Linus Torvalds has been quite clear on in-kernel localisation support. He says that localisation support can be given but not to do it in the kernel. Rather do it in klogd or similar.

Thus, Kernel localisation is a topic that is still under debate and not really taken up by anyone till date.

17.3. Linux Console Localisation

Linux Console localisation is also an issue being discussed and worked on. Linux Console localisation deals with writing/viewing localised text on the console. Currently, this supports only western languages. But there are discussions on the linux-utf8 mailing list to support more languages. The main idea is to avoid adding Linux console support in the kernel but rather add the support in a user-space software component. For further details refer [HKR4] and [HKR5].

Chapter 18. Tools for Localisation

In this chapter, we briefly describe a select set of software tools that are useful in various stages of localisation. These include tools for creating fonts, text editors which can handle Unicode text, translation aids, etc. If you find a tool relevant or useful, you can use the URL to access the tool and/or for more details on the tool. Most of these tools are also included in the CD associated with the handbook.

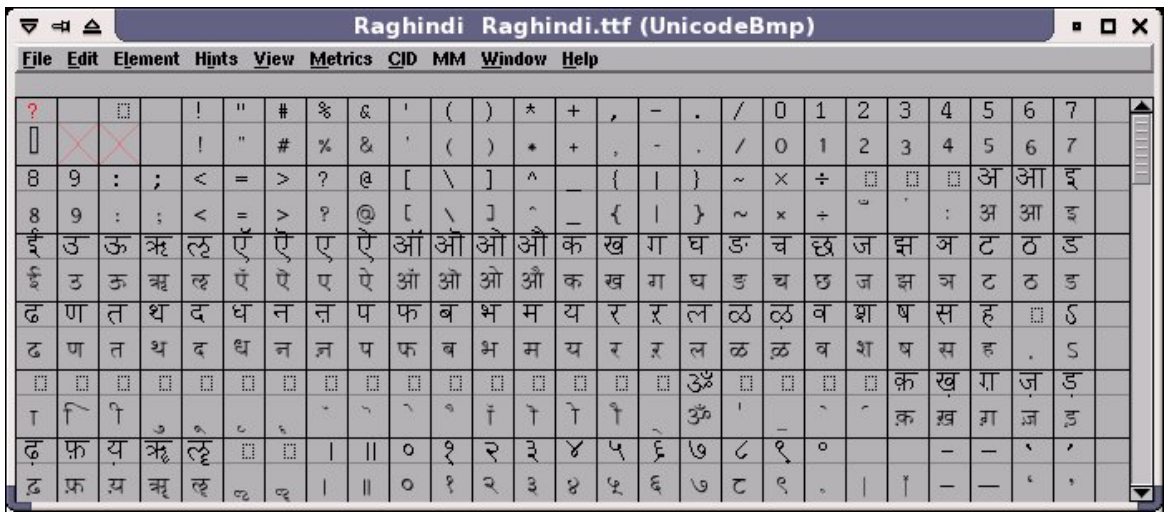
18.1. Tools for Fonts

This section describes some tools used to create fonts.

18.1.1. fontforge

Area: Font creation

Figure 18-1. A view of FontForge font editor showing Devanagari fonts, developed by CDAC (formerly NCST)



Description of the tool: Fontforge is a GUI tool which allows us to create and modify vectored and bit-mapped fonts (postscript,truetype and opentype fonts). The fonts can be saved in different outline formats, and generate bitmaps. The outline view is where you actually edit the splines that make up your glyphs. It allows us to create complex curves using GUI interface. There is also a birds eye view to all the characters in the font.

URL: http://fontforge.sourceforge.net/

Remarks: This tool gives lot of flexibility while designing a font. The help manual is quiet good. Font creation theories are covered in detail which will help a naive user

delve into the issues of font creation.

18.1.2. xmbdf

Area: Font creation

Description of the tool: xmbdf is a bdf (bitmap distribution format, see section Section 7.2.1) font creation tool. Multiple glyph bitmap fonts can be edited at the same time using this tool.

URL: http://linux.maruhn.com/sec/xmbdfed.html

18.1.3. gfonted

Area: Font creation

URL: http://www.levien.com/gfonted/

Remarks: Intended to become a font editor for Adobe Type 1 format fonts. It is currently in the very early stages of development.

18.1.4. ttf2bdf

Area: Font conversion

Description of the tool: This tool can be used to convert fonts in TrueType format to fonts in BDF format.

URL: http://crl.nmsu.edu/~mleisher/ttf2bdf.html

Remark: This tool uses FreeType TrueType rendering library to generate BDF bitmap fonts from TrueType outline fonts at different sizes and resolutions.

18.2. Editors

The editors included here are primarily from the perspective of editing Unicode text, which is required when editing PO files while inserting translation in your language.

18.2.1. Mozilla Composer

Area: Multilingual HTML Editor

Description of the tool: This HTML editor is provided along with the Mozilla suite. This tool supports a large number of encodings (including UTF-8). The edited file can be saved as text document in various encodings.

URL: http://www.mozilla.org/editor/

Remarks: Spell checker support for indic scripts are not available here (http://dictionaries.mozdev.org/installation.html) [DCT1] contains a list of dictionaries for some European languages which can be downloaded.

18.2.2. Yudit

Area: Multilingual Text Editor

Description of the tool: Yudit is a multilingual unicode text editor for the X Window System. The Menu translations available in 28 languages with FAQ support for 12 languages. One of the major features of this editor is that it does not depend on any external engine to render the font, print the text and transliterate keyboard input. Locale independent document in postscript format is generated to facilitate the printing support. The document can be saved with various Unicode encodings. It has bidirectional text support in Unicode and implements Unicode Bidirectional Algorithm UAX#9.

URLs: http://www.yudit.org/ and http://yudit.org/install.html

Remark: This editor has support for indic scripts like Tamil, Devanagari, Bengali, Gujarati, Gurumukhi, Oriya, Malayalam, Kannada and Telugu.

18.2.3. Simredo

Area: Multilingual Unicode Editor

Description of the tool: Simredo is a Java Unicode Editor and is freely available. It can Convert to and from many Unicode encodings. It has built in support to both RL(Right to Left) and LR (Left to Right) languages. A keymap function is provided which is flexible, which is used to redefine keyboard layout. It uses an inbuilt-encryption function. A new conversion function is added in order for the users to be able to define their own conversion tables.

URL: http://www4.vc-net.ne.jp/~klivo/sim/simeng.htm

Remarks: An Editor for plain texts only, which means it cannot contain information regarding text style, font sizes etc. Requires Java Runtime Environment (version 1.3 or higher) installed, and a sufficiently fast processor. The spell check features are available only for English and Esperanto texts. The editor is developed in Java and the source code is freely available. The spell checker code works with a dictionary.txt file which contains the list of lexicographically arranged words, each on its own line, which are compressed and sorted into a table in dictionary.dat.There are classes to even lexicographically arrange the words in dictionary.txt. The texts can be written from and to in both directions, so even Arabic texts can be included. The editor also has a window which shows all the character set in a particular font encoding.

18.2.4. Baraha

Area: Multilingual Word Processing tool supporting Transliteration rules.

Description of the tool: Baraha can be used to create documents in Indian languages.At present it supports Kannada, Sanskrit, Hindi and Marathi languages. You type roman texts (transliterated texts) which are mapped and converted to the letters of

corresponding languages .The documents can be saved in Unicode format. It can also be used as an Independent editor. An SDK for the Windows platform is provided to the users to create localised applications.

URL: http://www.baraha.com/index.htm

Remarks: Help is available in the form of transliteration rule for both Devanagari and Kannada. The text after these steps can can be exported in different formats(doc,html,txt,bmp,gif,png,jpg,pcx). This tool eases the typing from the keyboard. The program also comes with separate utility called BarahaDirect which can be used to type Kannada/Devanagari text directly into MS Word, PageMaker and other applications(using transliteration). The only requirement for the working of BarahaDirect is the Unicode support by the target application.

18.2.5. Emacs

Area: general purpose editor which can be used to edit PO files.

Description of the tool: Also known as real-time display editor, Emacs is supported under multiple platforms. The editor can be used to edit PO files (used in the translations) using the po-mode of the editor.

URL: http://ftp.gnu.org/pub/gnu/emacs/ and http://www.gnu.org/software/emacs/

18.3. Translation Tools

18.3.1. KBabel

Area: Editing and managing gettext PO files.

Description of the tool: It is a tool which comes with an editor, which can be used as dictionary for different languages. Full navigation capabilities, full editing functionality, possibility to search for translations in different dictionaries, spell and syntax checking, showing diffs, etc are some of the capabilities of the tool.

URL: http://i18n.kde.org/tools/kbabel/

18.3.2. Mozilla Translator

Area: Translation

Description of the tool: MozillaTranslator is for localisers/translators who want to localise Mozilla. This program is written in Java. It provides the interface for changing the strings and when done repackages the files for redistribution. This tool is available in both flavours X11 and Win32. The program is available on the official Mozilla Translator web site, as an executable Java ARchive. The Contributors are allowed to register and upload their localisations of mozilla.org related projects in the Mozilla Translator site.

URLs: http://www.mozillatranslator.org/ and
http://www.asturies.com/viesca/software/mozilla/MANUAL.HTM

18.3.3. gtranslator

Area: PO files Editor

Description of the tool: gtranslator is a GUI IDE/editor (on GNOME) for gettext PO files (used to translate messages in the form of "msgid"[original] and "msgstr"[translated]) . It offers easy way to edit PO files and its alike files (e.g. po.gz .mp/gmo). It allows users to easily navigate among fuzzy, translated and untranslated messages. The outdated PO files can be merged with a recent POT file. Features like find and replace are available. The editor also translates your PO files automatically, using the learn buffers and the gettext domain available.

URLs: http://gtranslator.sourceforge.net/ and
http://sourceforge.net/projects/gtranslator/

18.3.4. poedit

Area: PO files Editor

URL: poedit.sourceforge.net/, http://sourceforge.net/projects/poedit/

18.3.5. translate

Area: Online Project which has localisation tools.

Description of the tool: translate utility comes with a set of toolkit which help in the localisation process. There were mainly four tools used for localisation purposes,namely

1. oo2po:This tool converts OpenOffice.org specific GSI/SDF files to PO[1] files.
2. po2oo: This tool converts PO files back to the OpenOffice.org specific GSI/SDF files.
3. moz2po: This tool converts mozilla specific xml files to PO files.
4. po2moz: This tool converts PO files back to mozilla specific xml files

URL: http://translate.sourceforge.net/

Remark: Require Python script for translating to and from DTD files of Mozilla.

18.3.6. poxml

Area: Tranalations of

Description of the tool: poxml is a collaction of tools that are used to translate DocBook XML files, using PO files in gettext framework. There are mainly five tools

1. xml2pot: This tool creates a POT file from DocBook XML file. For example

   ```
   xml2pot original.xml >original.pot
   ```

   This command will create a POT file named tools.pot.

2. po2xml:This tool translates DocBook XML using a PO file. For example

   ```
   po2xml original.xml translated.po > translated.xml
   ```

   This command will merge translated strings of PO file with the original XML file which was intended to be translated.

3. split2po: This tool produces a PO file from two XML files (original and translated XML files). This PO file represents the changes between the two XML files. For example

   ```
   split2po original.xml translated.xml > difference.po
   ```

4. swappo: This tool swaps the msgid and msgstr fields of the PO file. For example

   ```
   swappo translated.po
   ```

   The command will swap the msgid and msgstr fields of the file translated.po.

5. transxx: e.g.

   ```
   transxx --text "nav" pofile.po > output.po
   ```

   The command will copy each msgid to msgstr and pad it with the word "nav" on both sides of the message.

URL: http://packages.debian.org/unstable/devel/poxml.html, http://weblogs.goshaky.com/weblogs/page/lars/20040823#translating_docbook_documents

18.3.7. Rosetta

Area: Web Based Translation Tool.

Description of the tool: Rosetta is a web based, Open Source, translation tool, which can be used host an online translation project for any Open Source Software Application.

URLs: https://launchpad.ubuntu.com/rosetta

18.3.8. Pootle

Area: Web Based PO file Translation Tool.

Description of the tool: Pootle is a web based, PO file translation tool, which can be used host an online translation project for any Open Source Software Application.

URLs: http://pootle.wordforge.org/, http://translate.sourceforge.net/

18.4. Miscellaneous

18.4.1. GNU FriBidi

Area: Implementation of Bi-Directional Algorithm.

Description of the tool: FriBidi is a free implementation of Unicode Bi-Directional Algorithm. Internally the library uses Unicode entirely.

URLs: http://fribidi.sourceforge.net/ and http://freedesktop.org/Software/FriBidi

Remark: API isn't complete yet.

18.4.2. protobidi

Area: Implementation of Bi-Directional Widget

Description of the tool: It is a prototype implementation of Bi-Directional Widgets,that has been implemented on top of a graphics library called "cnv". The application uses fribidi.

URL: http://imagic.weizmann.ac.il/~dov/Hebrew/protobidi/

Remarks: The tool is free.The application is still in its primitive stages.

18.5. Spell Checkers

18.5.1. ispell

Area: spell checker

Description of the tool: This program helps a user to find errors in the spellings in a file. There is a provision to provide users with a list of words which are the nearmost matches to the mis-spelled words.

URL:http://www.gnu.org/software/ispell/ispell.html
(http://www.gnu.org/software/ispell/ispell.html)

18.5.2. aspell

Area: spell checker library and independent utility.

Description of the tool: aspell is a better tool compared to ispell (described earlier). It can support multiple dictionaries at a time, with support for UTF-8. It will be a replacement for ispell tool.

URL:http://aspell.sourceforge.net/

18.5.3. dict

Area: client for DICT protocol

Description of the tool: DICT is a client for the DICT protocol(also known as Dictionary Server Protocol) defined in rfc2229.

URL:http://www.dict.org/rfc2229.txt,http://www.dict.org/bin/Dict

18.5.4. Spell

Area: spell checker

Description of the tool: spell is a tool which tries to emulate legacy UNIX spell tool.

URL:http://www.sunsite.ualberta.ca/Documentation/Gnu/spell-1.0/spell.html,http://linux.math.tifr.res.in/manuals/man/spell.html

18.6. Encoding Converters

Encoding converters are utilities which understand a variety of encodings. While there is overlap between the set of encodings known to each utility, there are also some encodings understood by one but not by the other, so there is a need for us to investigate these and pick one which suites our requirements. This section will not act as an exhaustive list. A more exhaustive list of Unicode enabled Products can be found here (http://www.unicode.org/onlinedat/products.html) [ENC1]. Some of the important encoding converters and libraries which help in the conversion process are listed below.

18.6.1. ICU

Area: library with Unicode support.

Description of the tool: ICU also known as International Components for Unicode is a Unicode framework which has a set of tools/utilities, libraries and API's which help in the conversion of encodings on most of the systems. This framework is an invaluable asset to those involved with character encodings in various ways. This library aims at the internationalisation of the applications by using UNICODE as de-facto standard to manipulate encoding conversions, transformation and other aspects related with encodings. It also has various tools like uconv (described later) to ease the command line manipulation of various encodings. The libraries are presently available in C/C++ and Java.

URL:http://oss.software.ibm.com/icu/, http://icu.sourceforge.net/, http://icu.sourceforge.net/userguide/localizing.html

Remarks:This framework deals with most of the issues involved with various encodings. This framework is also used by many popular software applications like Openoffice do deal with their encoding issues.

18.6.2. iconv

Area:character encoding converter

Description of the tool: iconv is a GNU libc tool, which converts the encoding of characters in a file to a different encoding. The result can be directed to another output or a different file. This tool has knowledge of a huge number of encoding which can be listed using the following command

```
iconv -l
```

iconv is used in the following manner to convert form one encoding to another:-

```
iconv -f from_encoding -t to_encoding [-o output_file] input_file
```

e.g.

```
iconv -f UTF-8 -t ISO8859-9 testin -o testout -c
```

URL:http://www.gnu.org/software/libiconv/

Remarks: Care must be taken by the user of this tool so that they do not attempt conversion between incompatible types. This utility can also be used to convert data (in form of text) from legacy applications,in old encoding formats, to Unicode or UTF-8 encoding formats.

18.6.3. uconv

Area: character encoding converter and transliteration tool

Description of the tool: uconv (part of IBM's ICU framework) converts the data from one encoding to another. This converter is similar to iconv (explained earlier) in every respect, with additional feature like rule based transliteration. The conversion process is achieved by converting the data to an intermediate encoding (i.e unicode encoding) and then to the target encoding. This tool can also be used for the transliteration purpose using the rules and syntax for text transformation (transliteration). This is achieved through a set of built-in composed transliterator or built in system transliterators, which are composed of strict ICU transliteration rules format. If the transliterator is not specified, transformation is achieved by specifying a set of rules adhering to ICU transliteration rules format. Overview of these rules can be obtained from the URL http://icu.sourceforge.net/userguide/TransformRule.html. This tool has knowledge of a huge number of encoding which can be listed using the following command

```
uconv -l
```

To convert form one encoding to another use the following command

```
uconv -f from_encoding -t to_encoding [-o output_file] input_file
```

e.g.

```
uconv -f UTF-8 -t UTF-16 testin -o testout
```

To use the text for the transliteration purposes it is invoked in the following manner

```
uconv -x [ transliteration ] [ Input file ]
```

Here the [transliteration] can be either a list of semicolon separated transliteration rules or a list of semicolon separated transliterator names.

How this transliteration can be achieved using the tool is demonstrated in the figure below.

Figure 18-2. helloworldcomt.c

In the figure, Latin-Devanagari and Devanagari-Latin are composed transliterators. The second command in the figure uses the rule based format to transliterate. More complex rules can also be written.

URL: http://www.uconv.com/,http://oss.software.ibm.com/icu/

Remarks:Because Unicode encoding is an intermediate step in the conversion process, this tool can well be used in the internationalisation process of applications.

18.6.4. uniconv

Area:character encoding converter

Description of the tool: uniconv converts the data from one encoding to another. This tool is distributed with multilingual editor Yudit and understands a wide range of encodings which can be listed using the following command.

```
uniconv --help
```

This command can be used in the following manner.

```
uniconv  -out  outpu-file  [-decode  inpt-encoding  ]-encode output-encoding ]
                                                                              [
```

e.g.

```
uniconv -decode  java  -encode  utf-8  inputfile  -out outputfile
```

URL:http://yudit.org/

18.6.5. convmv

Area: filename encoding converter

Description of the tool: convmv is a tool (GNU GPL) which converts filenames and directories from one encoding to another. This tool can be handy if there is a switch between one locale to another locale. This tool can be used in the following manner

```
convmv [options] FILE(S)
```

which expand as

```
convmv -f [from-encoding]  -t [to-encoding] .........  FILE(S)
```

e.g.

```
$> convmv --notest --upper testin
$> ls  TESTIN
TESTIN
$>
```

To know more about the options available invoke the command

```
convmv --help
```

URL: http://downloads-zdnet.com.com/Convmv/3000-2238_2-10305901.html,http://www.download.com/Convmv/3000-2238_4-10305901.html

Notes

1. PO files are used to translate strings

Chapter 19. How-To's

In this chapter, we provide a brief description of a few systems such as CVS, Yudit, etc. which are often used in localisation process. The systems included are:

- CVS: A version control system. Most open source software teams manage their software versions using this.
- Yudit - a Unicode based text editor.
- gtranslator - An IDE for editing and managing translations using PO files.
- KBabel - An IDE for editing and managing translations using PO files.

19.1. How-to use CVS

Read if: you are not aware of what is CVS and how-to use it

19.1.1. What is CVS?

CVS stands for Concurrent Versions System, a system which maintains the history of changes made to a set of files. This history will help the user of the file to keep track of all the changes made to the file. The CVS stamps each change made with the time it was made and the name of the user who made it. Usually, the user also provides information about why the changes were made. Thus the information described in the CVS helps the users find answers to the following questions:

- Who made what changes?
- Why were the changes made?
- When were the changes made?

More importantly multiple users can work on the system and make changes on the various files constituting the system. CVS provides a simple and powerful mechanism for managing a central repository of the various versions of files constituting various versions of the system.

Since the CVS stores the older files, there is always a backtracking possible if any error is detected in the current version. Thus the CVS gives you a safety net. In addition, it also allows to track the user who errored.

19.1.2. How to work with CVS

Individual users or developers can have a version control system running on their local machine. But people working in a team need a central server, which all members can

access, to serve as their central repository of files. In an office this can be done by having the repository on a server on the local network. For an open-source project, this model can be implemented using the Internet. CVS has built-in client-server access methods so that a user who can connect to the Internet can access files on a CVS server.

A user should check-out a file (from the repository) he wants to use, modify it and then check it back in. Checking-out a file gives the user exclusive rights on that file. This means that no user can access a file which has been checked out by someone else. This leads to some problems. For example, Aarti checks out 'hello.java' and goes for lunch, and suppose Indira is trying to rectify some bugs and realises that one of the bugs is in 'hello.java'. Now Indira cannot get access to the file till Aarti checks it back in. And so she will have to wait till Aarti returns from lunch and checks the file in. At the same time, this is precisely the strength of a version control system. Note that if Aarti and Indira made changes to the same file independently, one has to integrate these changes at the end to create the final file, meanwhile nither knows of the other's changes. Such integration is extremely complex in general and hence best avoided.

In a large open source project however, where there are many developers working late nights in any time zone, giving one user exclusive rights on a file and thus preventing other users from accessing the file will clearly not work. CVS handles this problem with its unreserved check-out model. Checking out a file does not give exclusive rights to a user. A file can be checked-out by multiple people at the same time, i.e., multiple people can modify the files at the same time and check it back in. CVS detects when multiple users are modifying the file and automatically merges the changes unless the changes are on the same line. If the changes are on the same line then the users have to manually update the file.

19.1.3. CVS commands

19.1.3.1. Setting your CVS repository

CVS records everyone's changes to a given project in a directory tree called the repository. Hence, before you can use the CVS, you are required to set the CVSROOT environment variable to the repository's path. Suppose the CVS repository is '/src/master', then the command you need to enter to set the CVSROOT variable if your shell is a csh or one of its descendant is

setenv CVSROOT /src/master

and if your shell is Bash or other Bourne shell variant the command is:
CVSROOT=/src/master

export CVSROOT

If you forget to do the above, CVS will complain if you try to run any CVS commands.

19.1.4. Checking out a working directory

All your files will be stored under a directory created by the CVS. You will need to check out this directory. For this purpose you use the cvs checkout command. For example, suppose your project name is 'l10n' then you use the following commands

cd

cvs checkout l10n

The above command means "Checkout the source tree called 'l10n' from the repository specified by the CVSROOT environment variable." All your project files will be created under this directory. There will also be a cvs directory created under this directory. CVS uses this directory to record extra information about all the files in the directory, to help it determine what are the changes made by each user since the time the files have been checked-out.

19.1.5. Making changes to a file

Once the CVS has created the directory for you, you can edit the files it contains in the usual way like you do in any other directory.

19.1.6. Merging your changes

Since each user of your team will be working on his own working directory, the changes made by one user will not be visible to others. For this purpose you need to commit your changes periodically, after you are done testing. But since it is also possible that there are other users who are working on the same file as you, you need to first update the main tree. This is done by the cvs update command. This command will essentially merge someone else's changes into your code. Hence it is best to to make sure that things still work.

19.1.7. Committing your changes

Having brought your sources up to date by merging them with the rest of the group and having tested them, you now need to commit your changes to make it visible to the rest of the group. You do this using the cvs commit. The syntax for this command is

cvs commit filename

where filename stands for the name of the file you have modified.

19.1.8. Examining changes made by others

If you want to know about the changes made by others in your group to a certain file, you use the cvs log filename command, where filename stands for the name of the file which you want to examine.

19.1.9. Adding a new file

CVS treats the adding and deleting of files like any other events and records them in files' histories. If you create a new file you will have to manually add it to the project using the cvs add command followed by the cvs commit command. The cvs add marks the file for adding. If you create a new file and do not mark it for adding and run the cvs update command then it will be flagged with a '?' character by which the CVS indicates that it does not know what it has to do with the file. Similarly you have to run the cvs commit in order to commit your changes and make it visible to all in the group

19.1.10. Deleting a file

Deleting a file is similar to adding. To delete a file from the project you first mark it with cvs rm and then do a cvs commit. Committing a file marked for deletion does not remove the files' history. It simply creates a new entry which is marked as "non-existent". The repository will still have records of the file's prior contents.

19.2. How-To use gtranslator

(Note:The purpose of this text is to demonstrate the use of gtranslator in translation using po files. We assume the reader has read the relevant section on the po files in the previous chapters and hence we will not explain the process in much detail.)

gtranslator is a GUI based PO-file editor, which can be used to translate messages in the PO files. The editor/IDE is still in the development stages and is going to evolve over the years. gtranslator eases the translation process by providing excellent translation management features. These features refrain translators from the technical aspects of the translation and enable them to concentrate more on the translation process (using PO files). Let us demonstrate the use of some of these features one by one.

The following aspect of gtranslator will be discussed in this section:-

- File formats
- Invoking gtranslator
- Navigation features
- Merging and updating PO files
- Autotranslate using translation Memory
- Compilation using 'msgfmt'

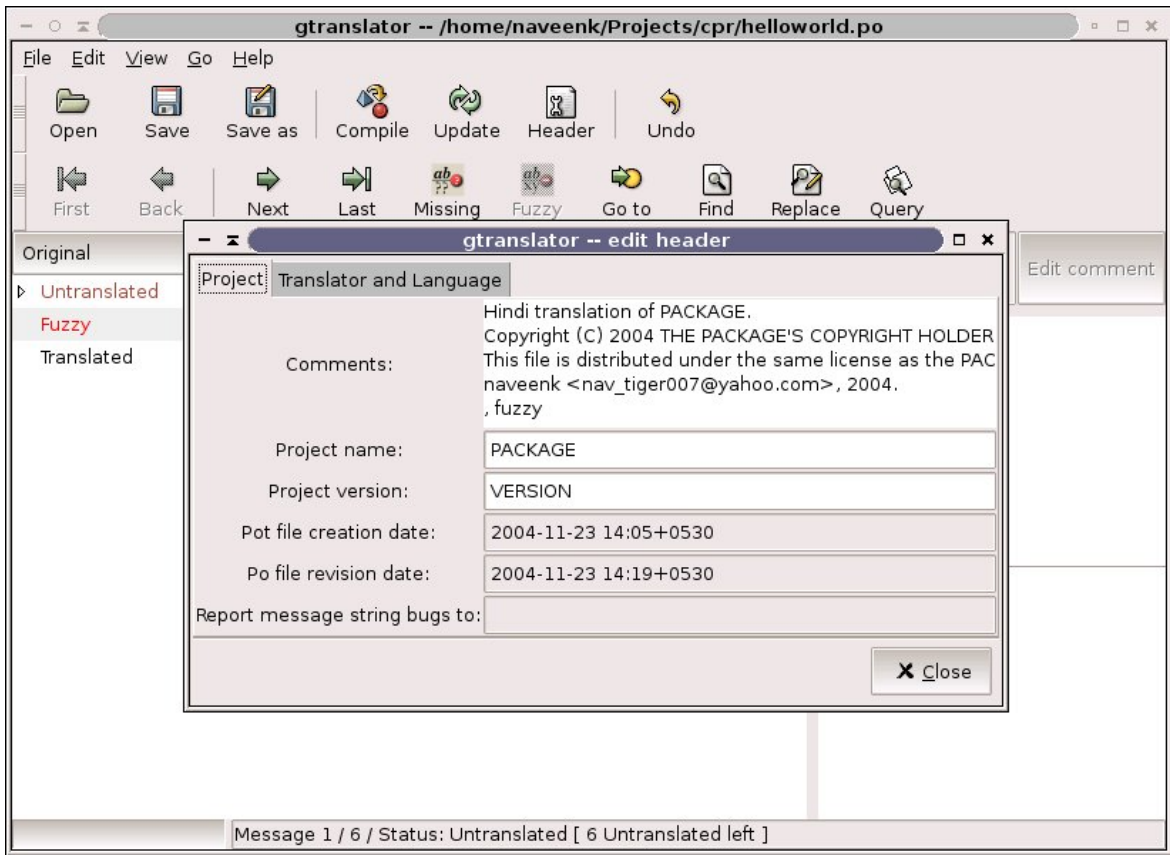
19.2.1. file formats

gtranslator supports .po files and its alike file formats like po.gz, .mp/gmo.

19.2.2. Invoking gtranslator

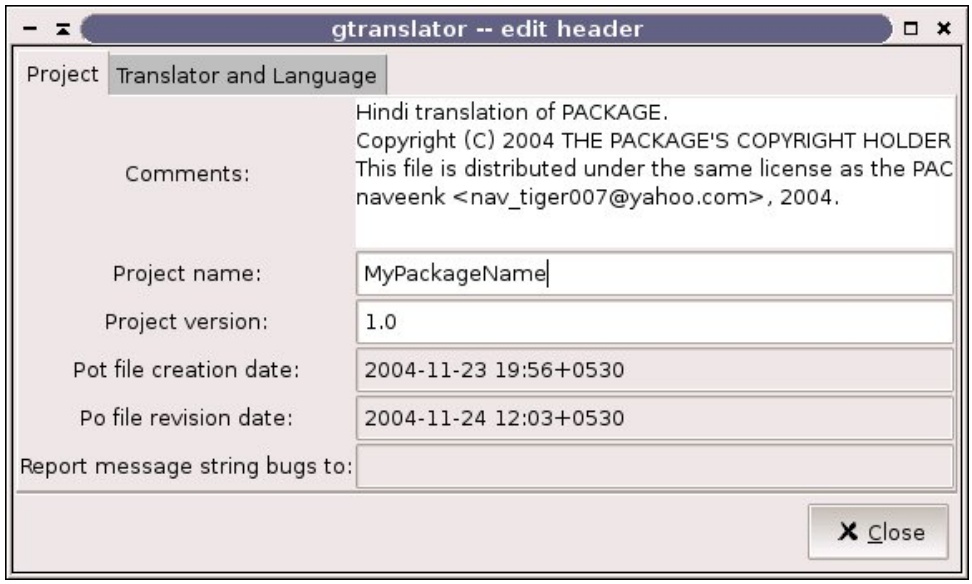
1. Invoke gtranslator by typing 'gtranslator' from the command line or select gtranslator from the programs menu
2. Open a PO file from it by clicking on the <open> tab on the menubar.Your screen will look something like Figure 19-1.

Figure 19-1. default view of gtranslator



3. Remove the word fuzzy from the Comments section in the edit header dialog box.
4. Replace VERSION with a numeric value in the Project version section of the edit header dialog box. This value identifies the version of the package
5. Remove PACKAGE and put your package name in the Project name section.

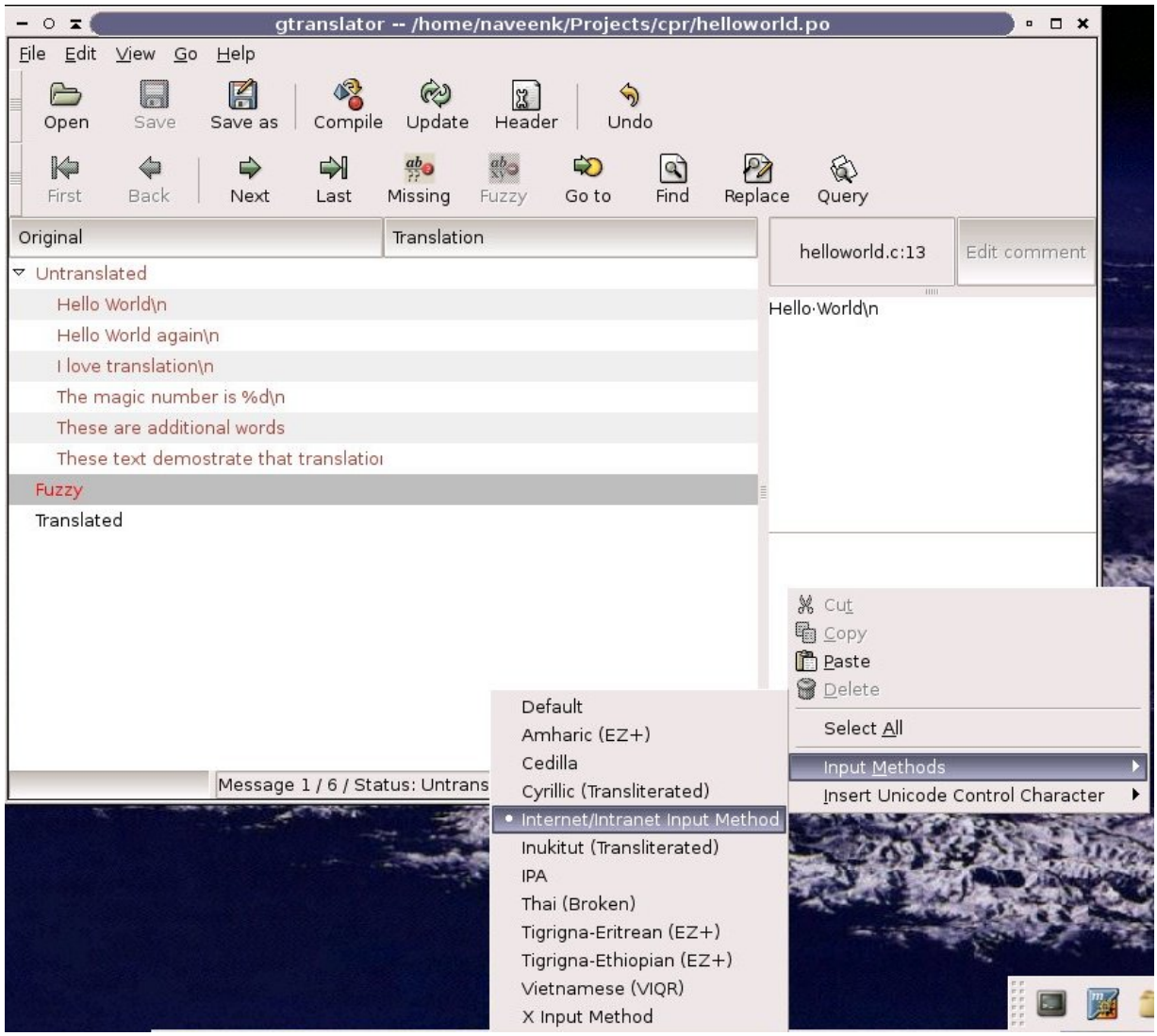
Figure 19-2. The gtranslator header section



6. Edit the header further according to your requirement if necessary; click <close> when done.

7. gtranslator gives us the multiple views of the PO file by categorising the messages into three categories as Untranslated, Translated and Fuzzy respectively as shown below in Figure 19-3

Figure 19-3. Selecting input method for gtranslator



8. Now you can translate the messages in your native language by selecting the appropriate input method (described in the section Chapter 9), in this case IIIM (described in Section 9.6.4). Figure 19-4 and Figure 19-5 are screen shots of such translation process.

Figure 19-4. translation process in gtranslator

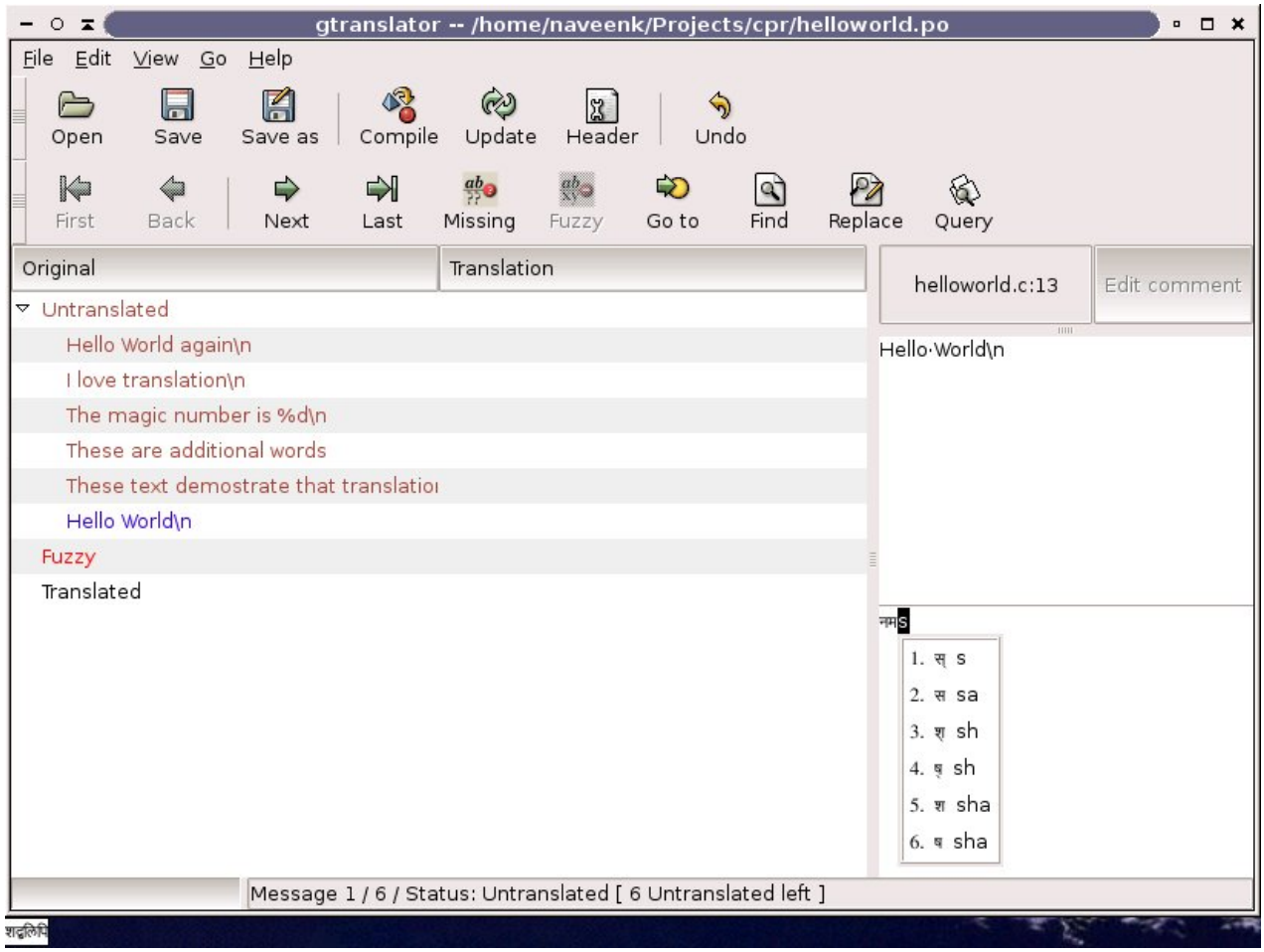
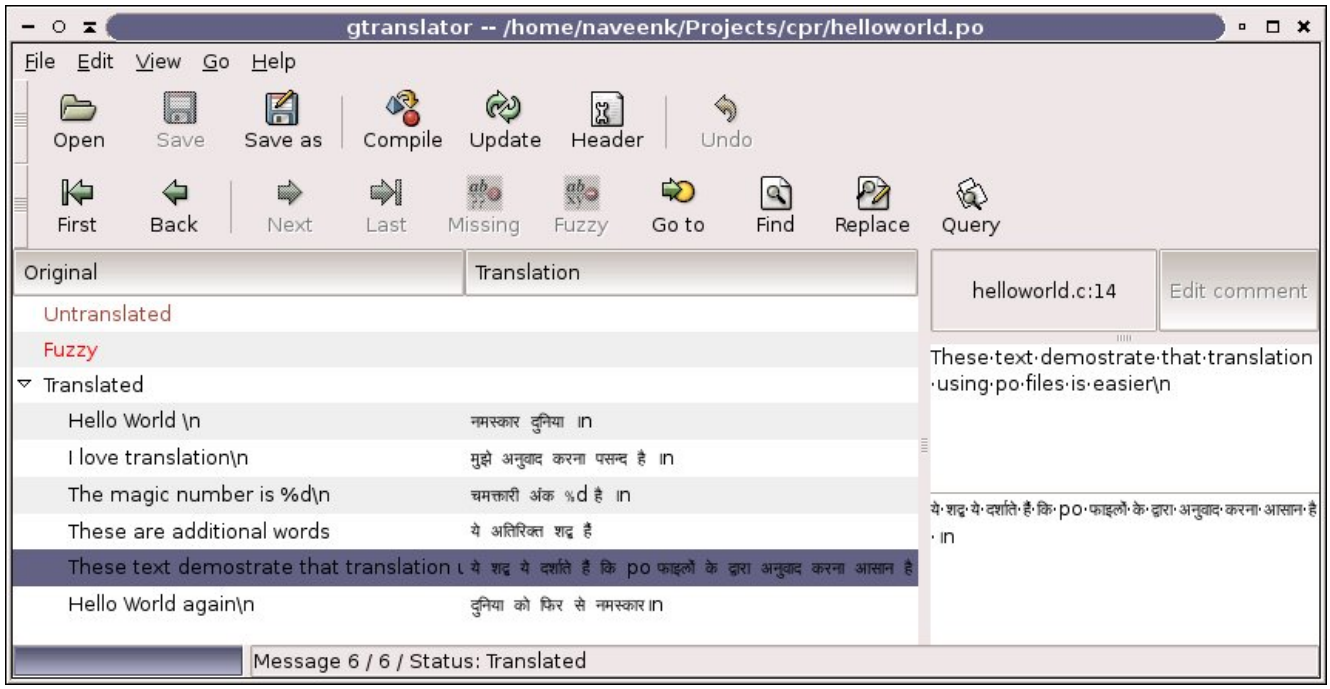


Figure 19-5. The translated string in gtranslator



19.2.3. Navigation features

The navigational feature in gtranslator allows to navigate between different kinds of translations (Untranslated/Translated/Fuzzy). The goto tab allows us to jump to any specified message number. Find and replace features allow us to find a string and replace a string. These features extend to both original and translated messages.

19.2.4. Autotranslate using translation Memory:

This feature is still in the development stage.

19.2.5. Compilation using msgfmt:

This feature is still in the development stage.

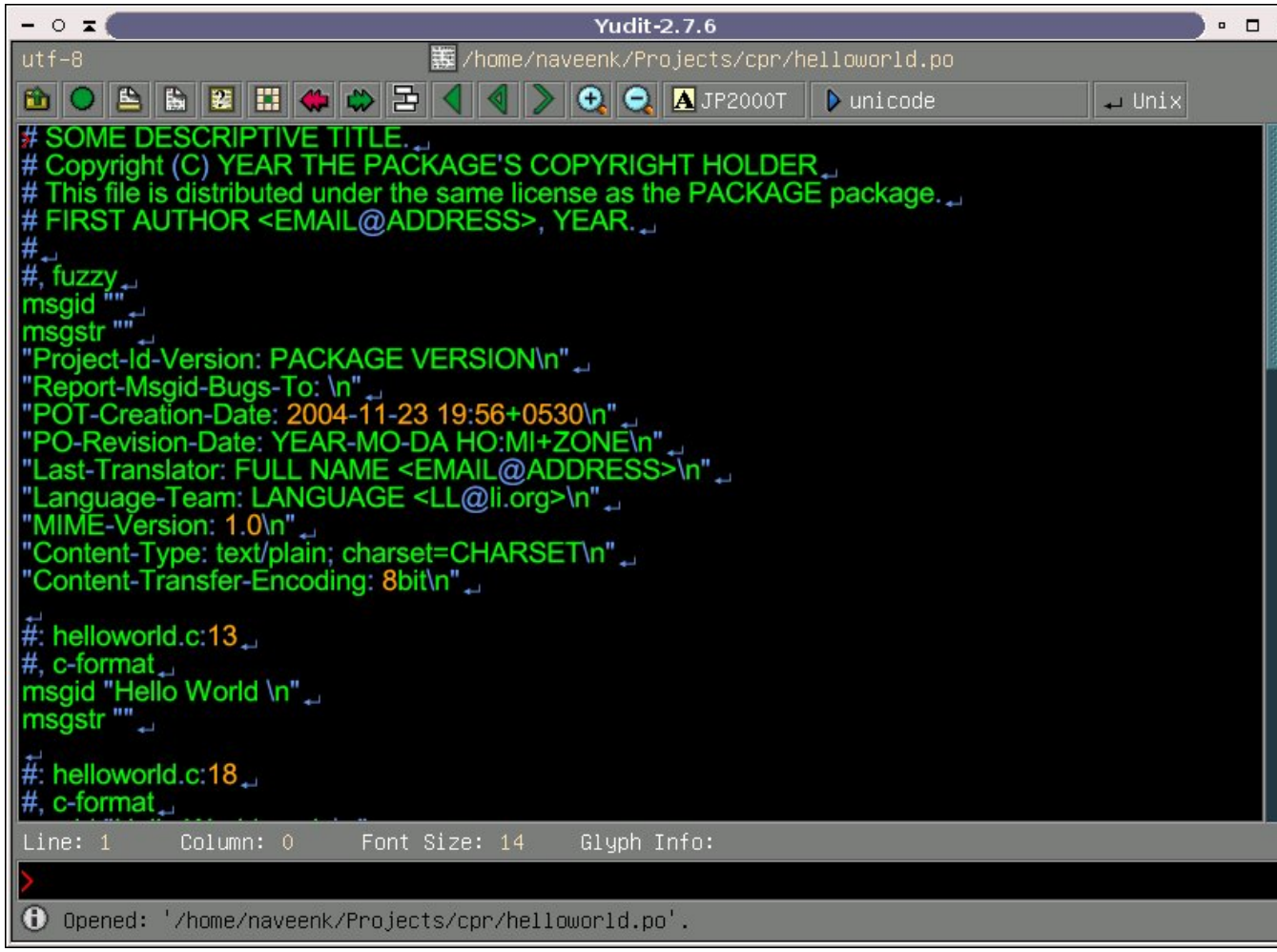
19.3. How-To use Yudit

(Note:The purpose of this text is to demonstrate the use of yudit in translation using po files. We assume the reader have read the relevant section on the po files in the previous chapters and hence we will not explain the process in much detail.)

Yudit is one of the simplest multilingual Unicode text editors which can be used to translate po files. The inscript and phonetic keymaps ease the task of typing in Yudit. The latest version of Yudit can be downloaded from here (http://www.yudit.org) [TYU2]. Now we will discuss how to translate po files using Yudit.

1. Invoke Yudit from the program menu and open a PO file by clicking on the open tab (see Figure 19-6).

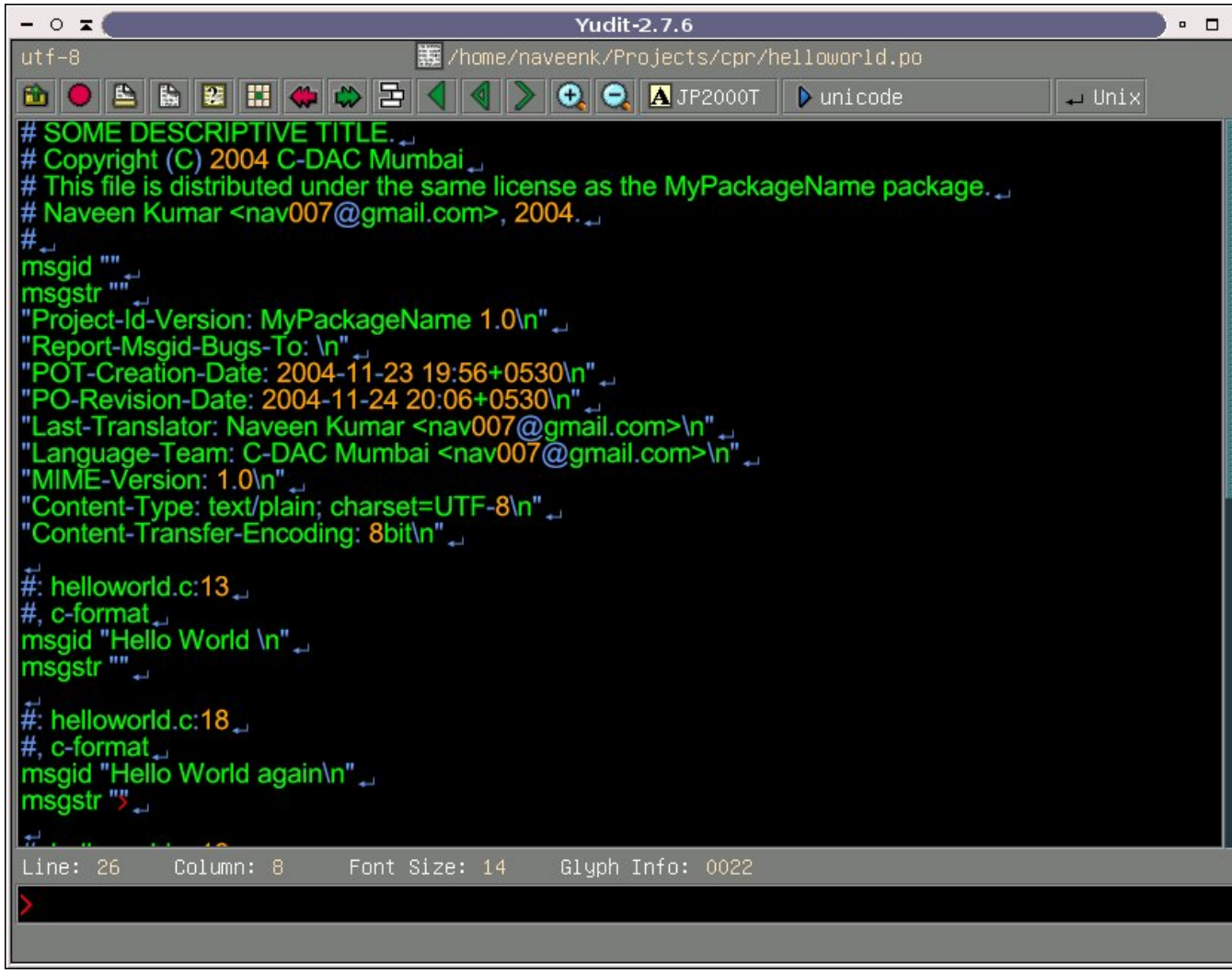
Figure 19-6. A view of unedited header information in Yudit



Now the header needs to be modified accordingly.

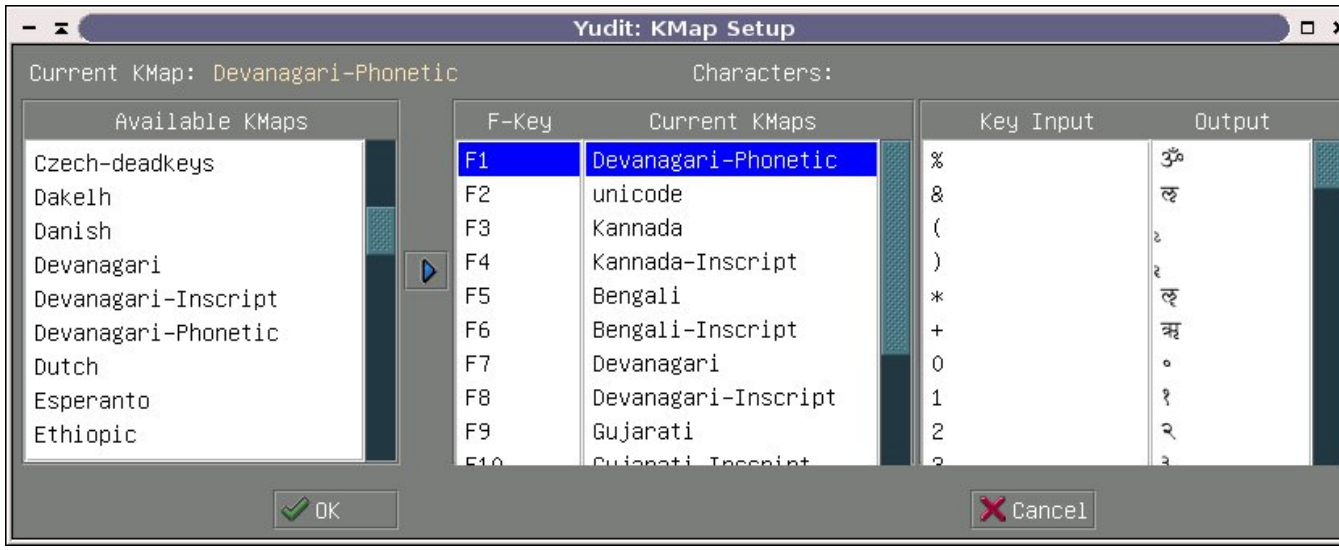
2. Before Starting the translation remove the line containing just the word fuzzy in the header.
3. Edit the line which contains Project-Id-Version. Put an appropriate package name and version.
4. Edit the line which contains Content-Type and Charset. Put UTF-8 in place of CHARSET.
5. Fill First Author, Last-Translator, Language-Team etc appropriately (Sample shown in the Figure 19-7).

Figure 19-7. A view of modified header indformation in Yudit



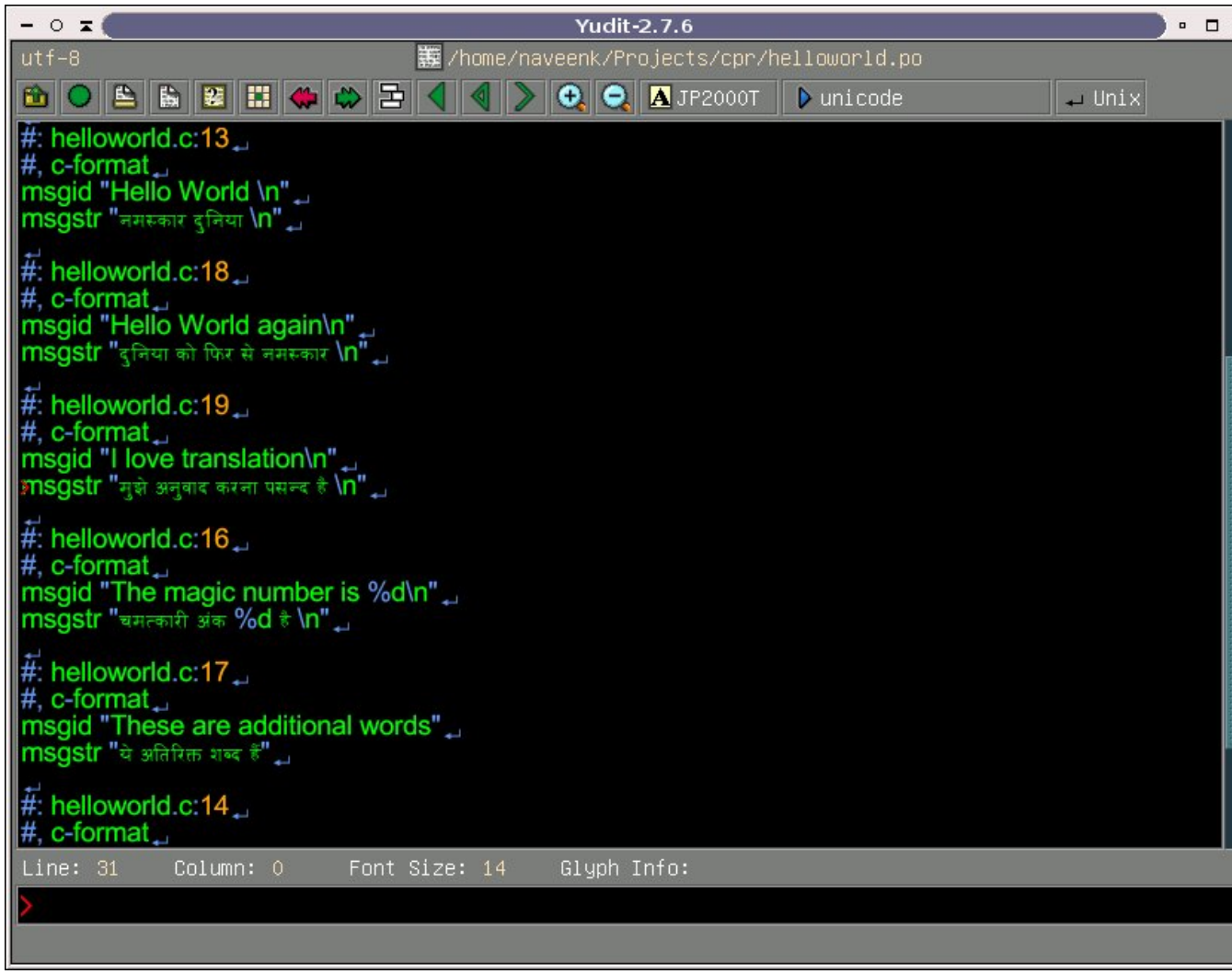
6. Now we are ready to translate the text. Select the appropriate Keymap.

Figure 19-8. Selecting a keymap in Yudit



7. We can use the keys from <F1> to <F12> to scroll among the input languages.
8. Select your language and start the translation in msgstr field.

Figure 19-9. A view of translated string in Yudit



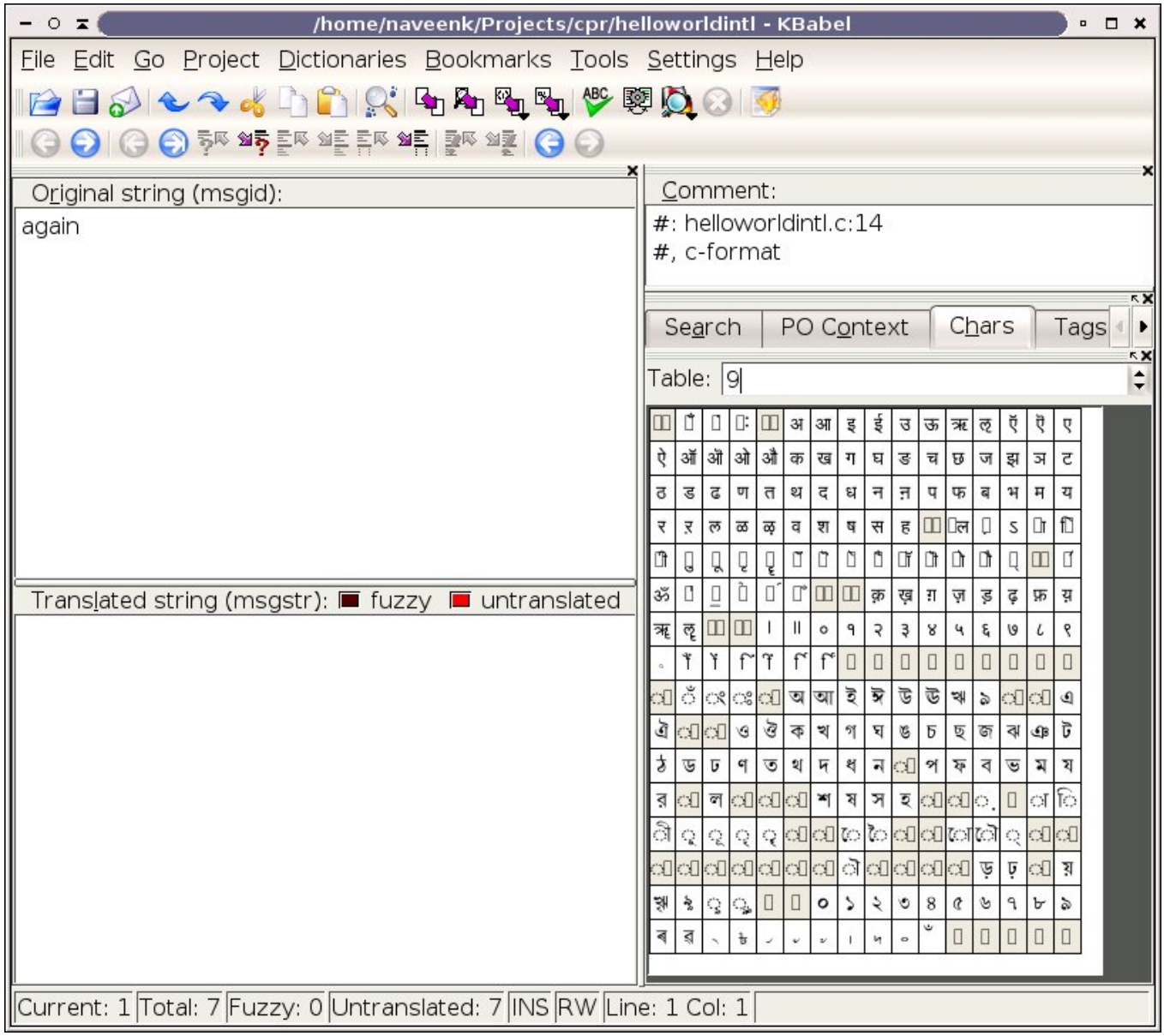
Although yudit helps in the translation process, one still need to depend on the gettext tools (e.g. msgmerge) for the management of translation. For more information on yudit please refer to yudit's website (http://www.yudit.org) [TYU2].

19.4. How-To use KBabel

(Note:The purpose of this text is to demonstrate the use of KBabel in translation using PO files. We assume the reader have read the relevant section on the po files in the previous chapters and hence we will not explain the process in much detail.)

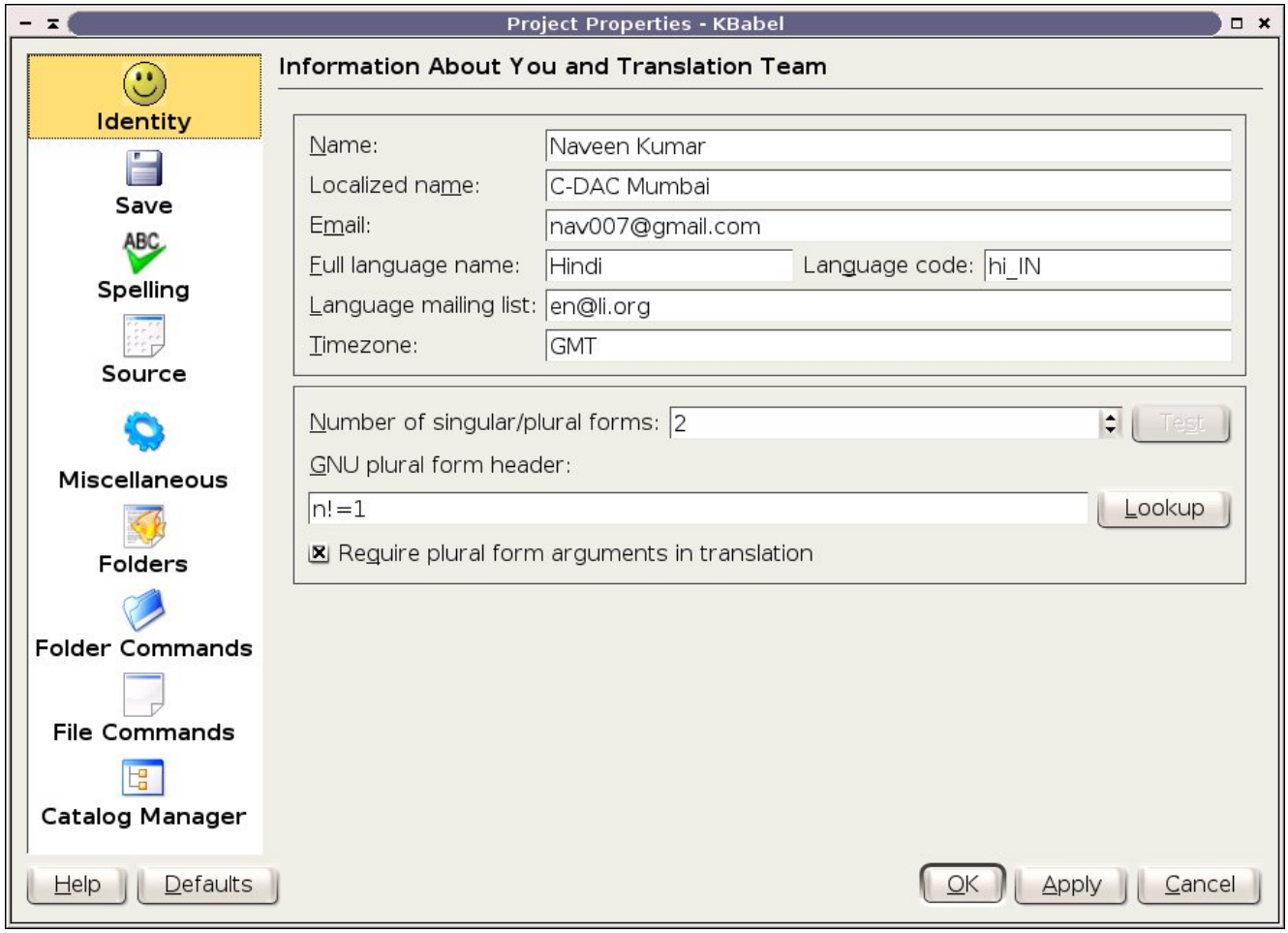
KBabel is one of the best editors available for the translation of PO files. The translation management features ease the use of this editor to a very large extent. This tool emulates translation memory with its dictionary feature. The users can auto-translate using these dictionaries. We will now demonstrate how to use KBabel to translate po files.

Figure 19-10. KBabel default view



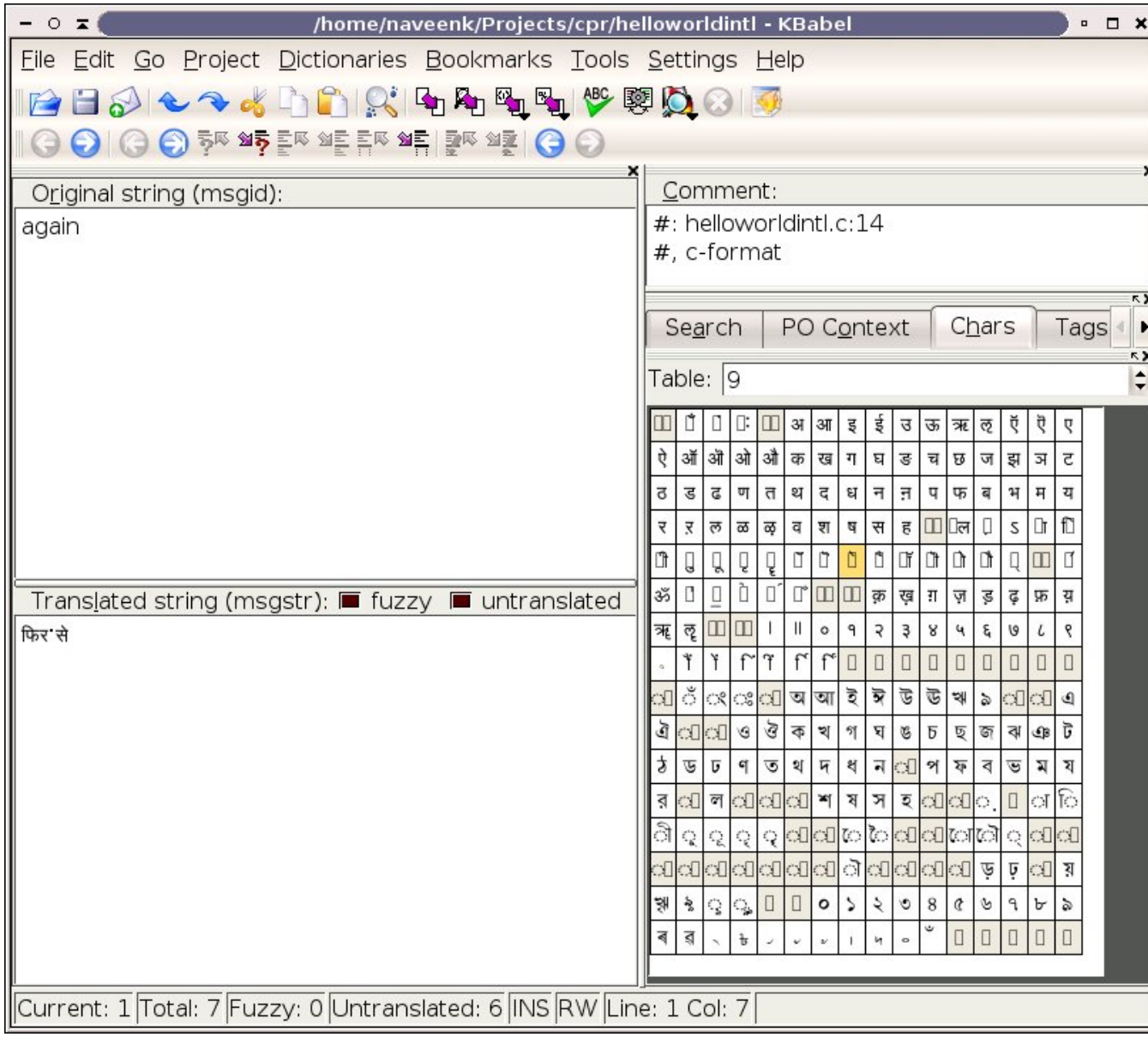
1. Invoke kbabel from the program menu and open a po file by clicking on the open tab.

Figure 19-11. Project properties



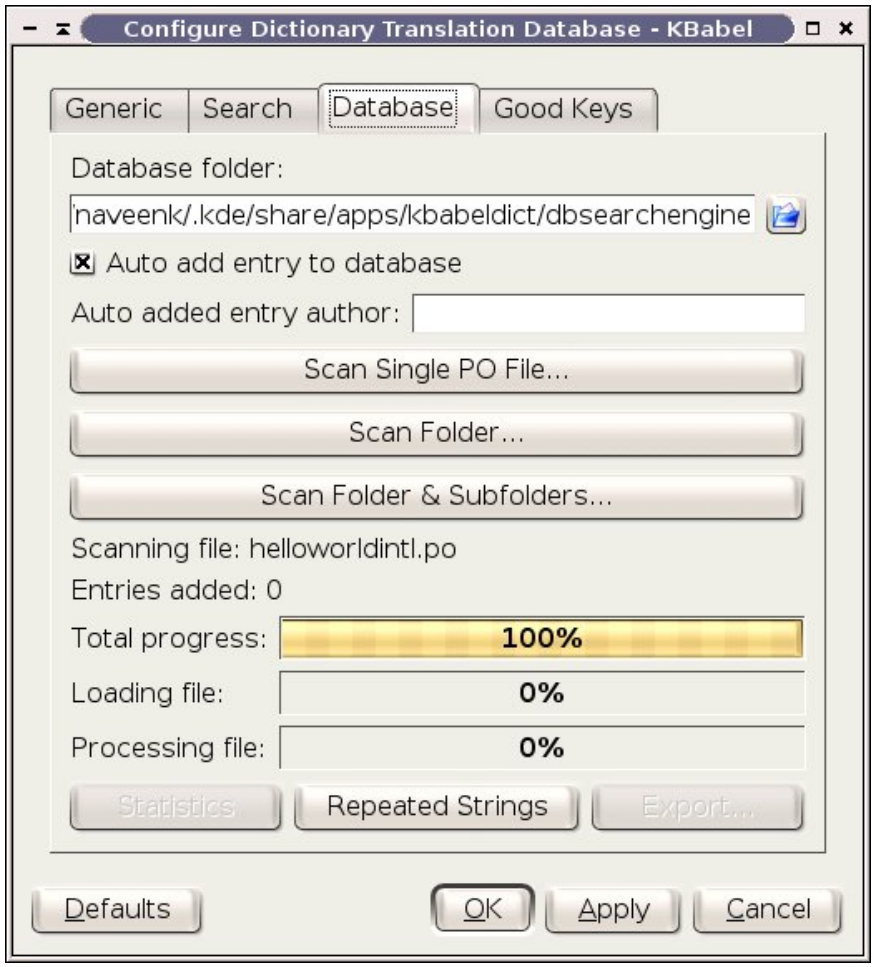
2. Open Project|Configure. Edit Project Properties window suitably(see Figure 19-11).Most of these properties will be reflected in the header section of the PO file.

Figure 19-12. translated string in kbabel



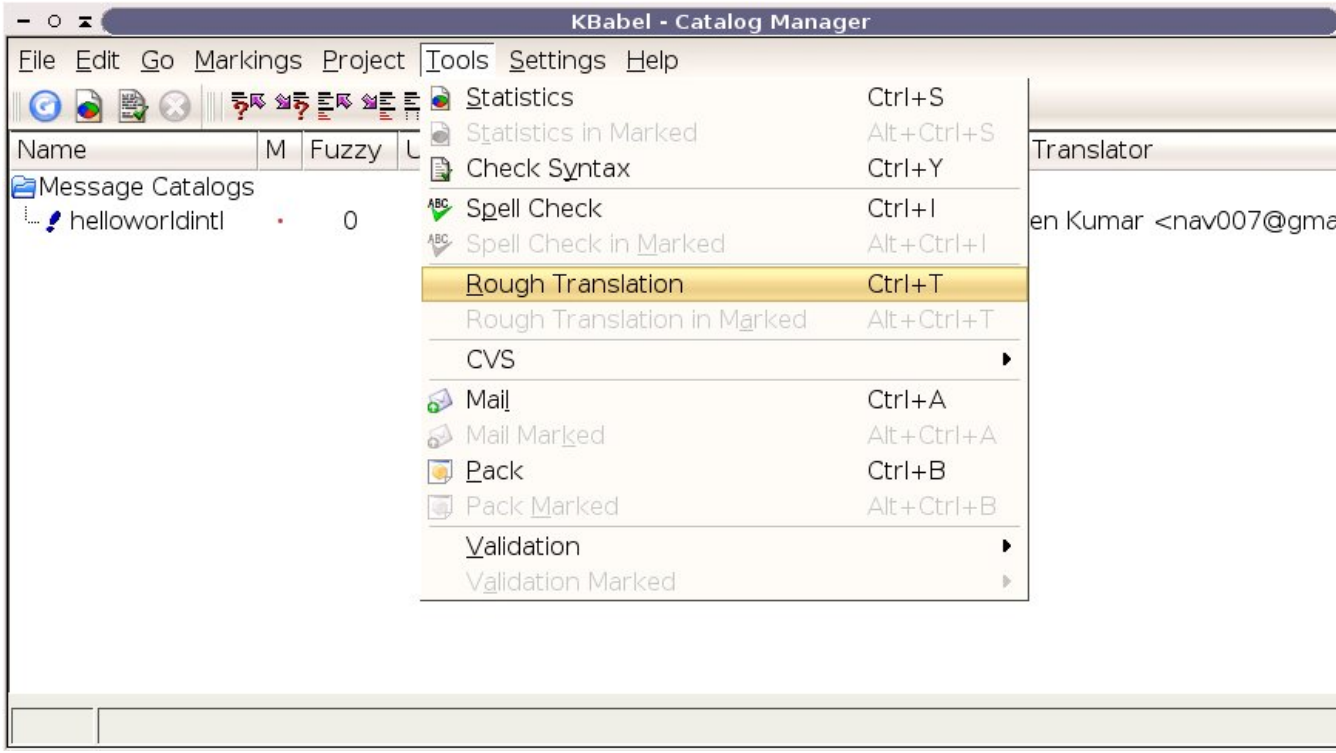
3. Translate the strings one by one (You can navigate the strings using the right arrow and left arrow buttons located in the upper left corner of Figure 19-12). You can use the Unicode table provided in the lower right section of the kbabel window.In this case the Unicode page for Hindi is located in table 9 (Unicode for Hindi starts from hex value of 0901). Entering any number in the table identifier brings up the relevant part of the table (Hindi in this case).
4. Edit the comment section if required (e.g. remove fuzzy tag).

Figure 19-13. Dictionary translation database window



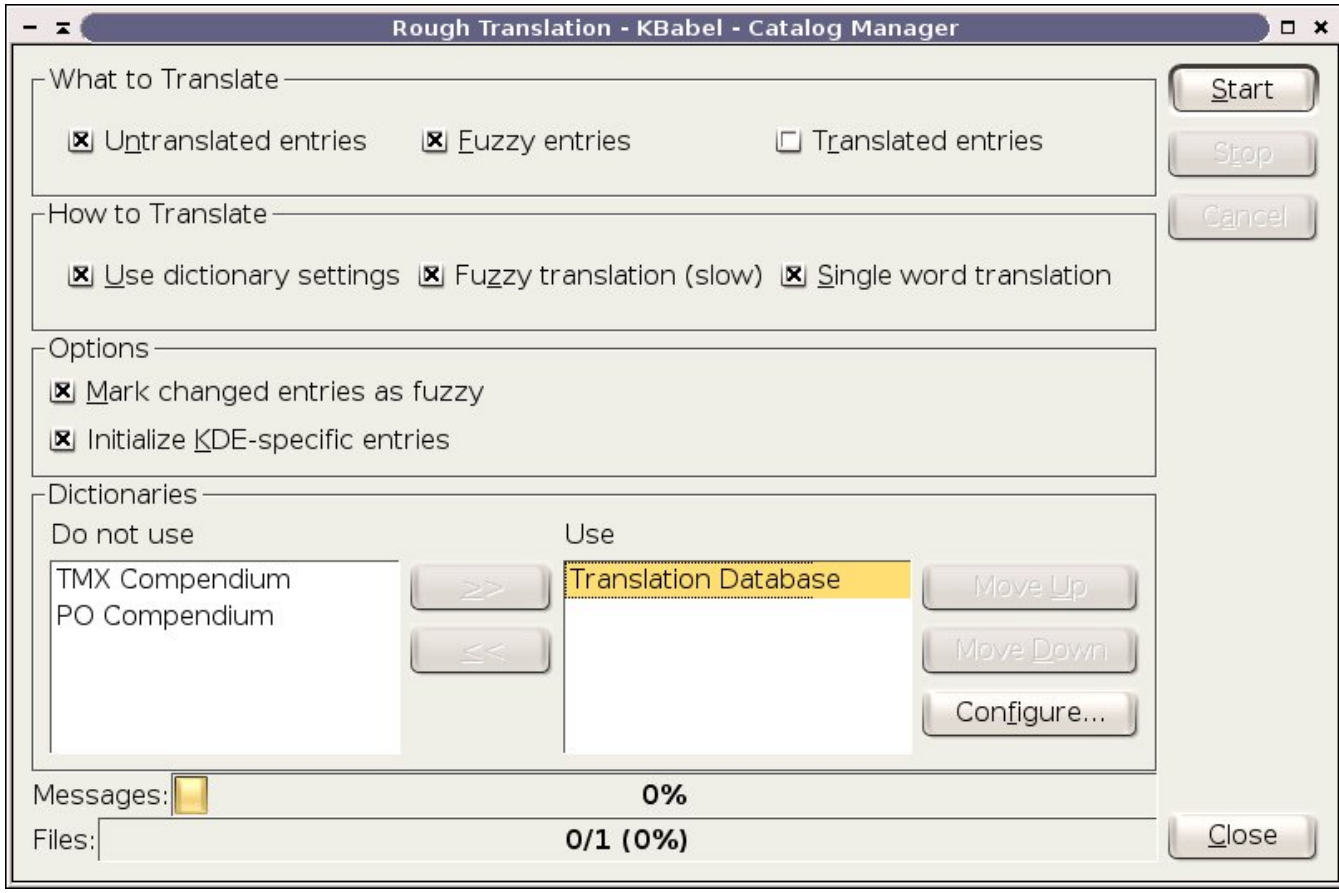
5. In order to use the directory facility, invoke Kbabel Dictionary and scan the folder which contains PO files of your application. All the existing translations will be added to the dictionary. These translations will be helpful in automatically translating the updated and new strings.

Figure 19-14. KBabel Catalog Manager



6. Invoke kbabel Catalog Manager and select Tools|Rough Translation. This will pop up a window as shown in the Figure 19-15

Figure 19-15. Rough Translation window in Catalog manager



7. Select the configure button in Figure 19-15 and select the appropriate Matching Method as shown in Figure 19-16. Change anything else, if required and click Apply and then OK. This will bring you back to the window in Figure 19-15

Figure 19-16. Configure Dictionary Translation Database window

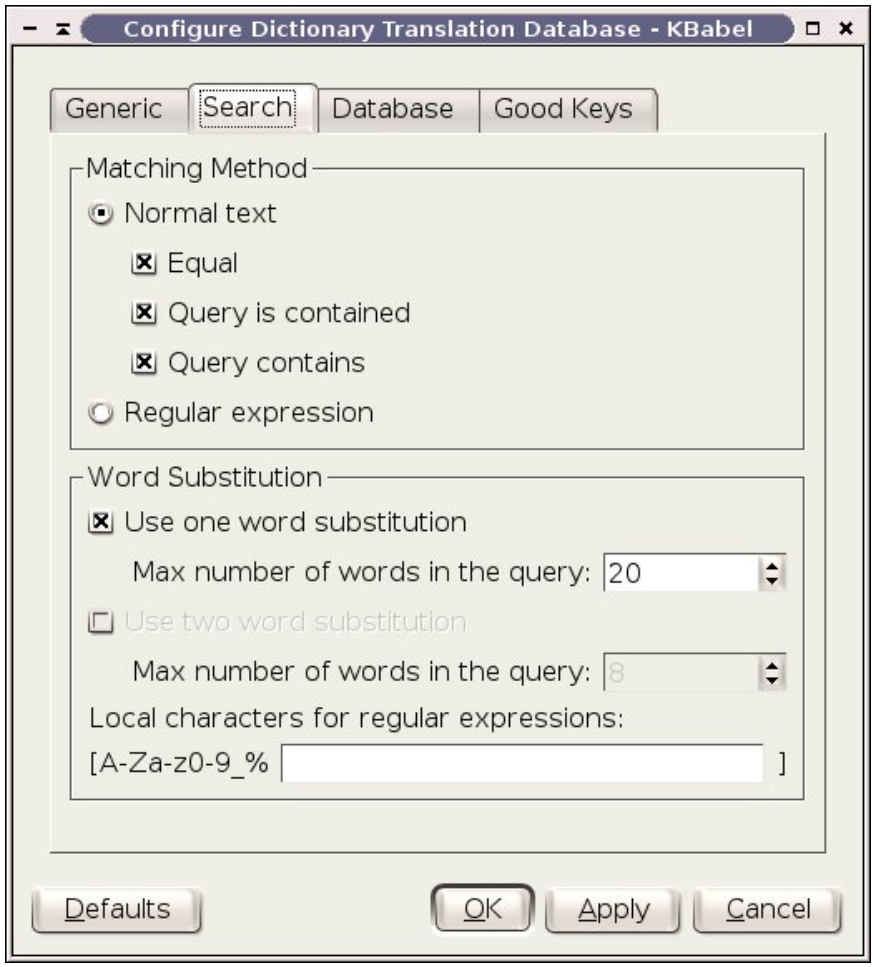
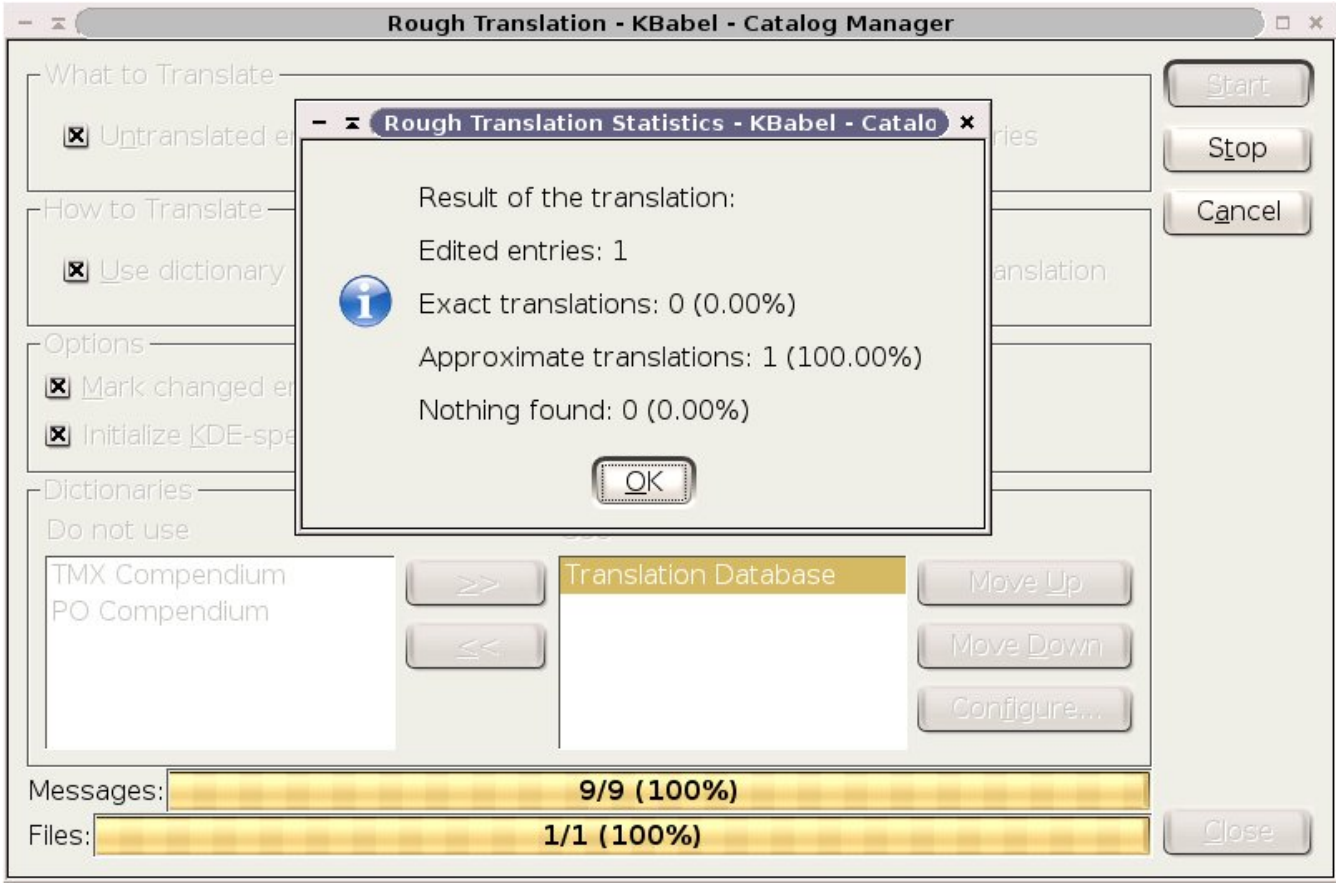


Figure 19-17. Rough Translation Statistics window



8. You are now back at the screen in Figure 19-15. Click on the start button to initiate the automatic translation. After the process ends a pop-up window shows the statistics of the translation process (as shown in Figure 19-17).
9. All the strings which are auto-translated are marked as fuzzy; the users need to manually check them before committing these translations.

The above mentioned auto-translation feature can also be used as word to word glossary based translation. Simply follow the following steps. (Note: These are not standard methods, the author came across this while experimenting with KBabel. Therefore we claim no responsibility of it's effectiveness).

1. Create a file (can be a file in C language), and lexicographically add words that may appear in the application string. Use gettext functions to mark the strings.
2. Extract the marked strings, in a pot file. Create po file, translate it word by word.
3. Add the po file translations in the kbabel database. Your dictionary is ready.
4. Now when you need to translate the PO file of the main application, simply invoke the auto-translation feature. This will perform word to word translation of the

strings. These words are replaced for their corresponding translations which was already present in translation database.

Chapter 20. Spell Checker

20.1. Introduction

(Note: In this chapter we will discuss the process of the creation of a spell checker for your language (hypolan) using, a generic spell checker tool like aspell.)

Aspell is a genieric Free and Open Source Spell checker engine, which can be used to implement spell checker in your language. This tool/engine can be used as a library if required. Otherwise this is used as an independent spell checker.

There are various algorithms that aspell engine follows to check for the misspelled words. These can range from replacement list to phonetic list. In an interactive mode on encountering a spelling mistake or a misspelled word in a file (see aspell manual page to learn various ways to use it), aspell provides the users with a suggestion list (which is a list of possible correct words for that misspelled word based on typing error or phonetic error). This suggestion list can be provided even when the spelling is correct. This is because the users might the type the words unknown in the aspell dictionary (or sounds similar to some of the words). Of course this is done taking into the account how frequently, most users, do these kinds of mistakes.

For example invoke the following command (in English locale)

```
prompt#aspell --lang=en pipe
```

This command will take you in an interactive mode where you can test your spellings by typing the words. If the words are incorrect aspell provides the users with a list of suggested words.

As another example invoke the following command (in English locale)

```
prompt#aspell --lang=en soundslike
```

This command will also take you in an interactive mode where you can check the phonetic equivalent defined by the language. The language specific phonetic codes are kept in a file which has the the following naming convention.

<name>_phonetic.dat

Aspell selects the appropriate dictionary (for a language) based on some options/environment variables or the current locale like LC_MESSAGES. In the next section we will discuss how to create a spell checker dictionary for your language.

Now in order to see the list of available dictionary use the following command

```
prompt#aspell dump dicts
```

If the dictionary for your language is not there maybe it's time to create one.

20.2. Creating dictionary

There are certain issues that you must take into consideration when you are creating a dictionary for your language. Some of these are.

1. How To deal with variations in words(affix variations, which collectively refers to prefix or postfix variations in a word). For e.g. occupy, pre-occupy, occupied, ocupies, occupying etc.
2. How to deal with common typing errors which occur most frequently in words in a particular language. For e.g the is written as teh.
3. What is the encoding that must be used for our language.

Let us discuss how can we solve these issues while creating a spell checker dictionary for a language.

20.2.1. The language data file

The language file is used to specify the relevent/necessary information or behaviour exibited by the spell checker for that language. The naming convention of the file has following format.

<ISO 639 Language code>.dat

Given below is the sample of such file for the English language.

```
name en
charset iso8859-1
special ' -*-
soundslike en
affix en
#repl-table en_affix.dat
```

This file has two mandatary fields (name and charset) and other fields are optional. If you language does not have this file you need need to create one for your file. More information on this file format can be obtained from the online manaul at URL http://aspell.sourceforge.net/man-html/The-Language-Data-File.html#The Language Data File (http://aspell.sourceforge.net/man-html/The-Language-Data-File.html#The Language DataFile)

20.2.2. Characterset file

This file is specific to a particular characterset. If the characterset for your language does not exist you have to create one. More information on this file format can be obtained from the online manaul at URL http://aspell.sourceforge.net/man-html/Creating-A-New-Character-Set.html#Creating A New Character Set

20.2.3. The Phonetic data file

This file specifies the phonetic transformation rules that can be specified for a language. This file can be specified in the optional field soundslike of the language data file. This file has the following naming convention.

<language_name>_phonetic.dat

Generally it is seen in many cases that even though the words are spelled incorrectly, they are phonetically the same. Hence this file is used to make suggestions based on the phonetic sound of the word. If you want to make use of the phonetic qualities of your language, you need to create this file. Thorough investigations must be carried out on the language before any such phonetic rules are made. For consider a sample given below for the english language.

```
AH(AEIOUY)-^          *H
AR(AEIOUY)-^          *R
A(HR)^                *
A^                    *
AH(AEIOUY)-           H
AR(AEIOUY)-           R
A(HR)                 _
BB-                   _
B                     B
CQ-                   _
CIA                   X
CH                    X
C(EIY)-               S
CK                    K
COUGH^                KF
CC<                   C
C                     K
DG(EIY)               K
DD-                   _
D                     T
EH(AEIOUY)-^          *H
ER(AEIOUY)-^          *R
E(HR)^                *
ENOUGH^$              *NF
-----------------------
-----------------------
-----------------------
X^                    S
X                     KS
Y(AEIOU)-             Y
ZZ-                   _
Z                     S
```

More comprehensive information related to the creation of this file can be obtained from the manual at URL
http://aspell.sourceforge.net/man-html/Phonetic-Code.html#Phonetic Code

20.2.4. The keyboard data file

The keyboards definition file is used to make the spell checker engine aware of the layout of the adjacent letters on the keyboard. This can be used by the spell chacker engine in determining the typing errors in proximity of the desired words.For example the word the can be miss-typed as thw Various algorithms make use of this file while doing the typo analysis in different ways. This file has the following naming convention

<keyboard layout name>.kbd

This file can be specified in the optional field keyboard of the language data file. And this file should be encoded in the UTF-8 encoding. A sample of the default sample.kbd file is given below which can be replaced by files like dvorak.kbd, qwerty.kbd etc.

```
qw
we
er
rt
ty
yu
ui
io
op
as
sd
df
fg
gh
hj
jk
kl
zx
xc
cv
vb
bn
nm
```

More comprehensive information related to the creation of this file can be obtained from the manual at URL
http://aspell.sourceforge.net/man-html/Notes-on-Typo-Analysis.html#Notes on Typo-Analysis

20.2.5. The affix file

Affix files are used to account for prefix and postfix variations in your language language. This file can be specified in the optional field affix of the language data file. An additional boolean field affix-compress can be specified to enable or disable affix-compression for faster retrieval. If your language does not have this file you need to create one so that variations based on grammatical rules can be easily accumulated. This file has the following naming convention

<language name>_affix.dat

A sample of such an affix file is given below for English language.

```
---------------------------
---------------------------

SFX V N 2
SFX V   e     ive        e
SFX V   0     ive        [^e]

SFX N Y 3
SFX N   e     ion        e
SFX N   y     ication    y
SFX N   0     en         [^ey]

SFX X Y 3
SFX X   e     ions       e
SFX X   y     ications   y
SFX X   0     ens        [^ey]

-----------------------------
-----------------------------

SFX P Y 3
SFX P   y     iness      [^aeiou]y
SFX P   0     ness       [aeiou]y
SFX P   0     ness       [^y]

SFX M Y 1
SFX M   0     ’s         .

SFX B Y 3
SFX B   0     able       [^aeiou]
SFX B   0     able       ee
SFX B   e     able       [^aeiou]e

SFX L Y 1
SFX L   0     ment       .
```

More comprehensive information related to the creation of this file can be obtained from the manual at URL
http://aspell.sourceforge.net/man-html/Affix-Compression.html#Affix Compression

Sometimes the replacement tables are also included in this files. These replacement tables are used when phonetic codes are not used.

20.3. Create the wordlist

After creating the above mentioned files you needs to create dictionary for your language. Create a list of words, in a file, seperated by a whitespace and then create then invoke the command like this.

```
prompt# aspell --lang=lang create master ./base < wordlist
```

Where base is the name of the file which contains the list of words.

More information on this can be obtained from the manual at URL
http://aspell.sourceforge.net/man-html/Creating-an-Individual-Word-List.html#Creating an Individual Word
List

Chapter 21. Resources

21.1. Fonts and encoding

[BAF1] http://www.nongnu.org/freebangfont/

- Bengali fonts available for download here.

[ENT1] http://www.joelonsoftware.com/articles/Unicode.html

- A must-read about Unicode and character sets.

[ENT2] http://www.utoronto.ca/ian/books/html4ed/appa/appa.html

- Information on character encoding)

[ENT3] http://www.unicode.org/unicode/reports/tr17/

- A good reference material on character encoding and Unicode.

[ENT4] http://www.unicode.org/charts/PDF/U0980.pdf

- Bengali to unicode encoding chart

[ENT5] http://www.microsoft.com/typography/otfntdev/encoding.htm

- A document which describes the encoding feature of open type fonts.

[ENT6] www.unicode.org/charts/

- This page contains links to the glyph charts and code points of characters of various languages.

[ENT7] http://www.w3.org/TR/charmod-norm/
(http://www.w3.org/TR/charmod-norm/)

- Reference for Normalisation to improve inter-operable text manipulation on World Wide Web.

[ENT8] http://www.joconner.com/javai18n/articles/UTF8.html#_ftn1

- An article on UTF-8

[ENT9] http://www.roguewave.com/support/docs/sourcepro/stdlibug/23-3.html

- A document on Character Encodings. Talks about Multi-byte encodings, Wide characters, Conversion between Multi-byte and Wide Characters.

[ENT10] http://std.dkuug.dk/jtc1/sc2/wg2/docs/n2176r.pdf

- A good piece on Normalisation and its implications on Character Encoding.

[ENT11] http://www.unicode.org/unicode/reports/tr15/

- Unicode normalisation Forms.

[ENT12] http://www.cl.cam.ac.uk/~mgk25/unicode.html#utf-8

- A FAQ on encodings

[ENT13] http://www.unicode.org/faq/unicode_iso.html

- Unicode and ISO 10646 FAQ

[ENT14] http://www.sensi.org/~alec/man/man_b/p_charset.html

- Portable Character Set

[FOT1] http://indic-computing.sourceforge.net/handbook/tutorial-fonts.html

- Tutorial on fonts (In progress)

[FOT2] http://www.microsoft.com/typography/specs/default.htm

- This document talks about the Open type specification. It is a very rich document with useful information for creating fonts in various languages with stress on Indic languages.

[FOT3] http://www.indlinux.org/wiki/index.php/FontInstallation

- This talks about how to install fonts in X Windows

[FOT4] http://www.microsoft.com/typography/WhatIsTrueType.mspx

- Information on true type fonts.

[FOT5] http://www.mozilla.org/projects/intl/fonts.html

- This document discusses the design and implementation of Mozilla's font subsystem.

[FOT6] http://www.tldp.org/HOWTO/Font-HOWTO/

- Font How-To by Donovan Rebbechi

[FOT7] http://en.tldp.org/HOWTO/FDU/index.html

- Font Deuglification How-To by Hal Burgiss

[FOT8] http://nwalsh.com/comp.fonts/FAQ/

- Font Faq

[FOT9] http://www.microsoft.com/typography/default.mspx

- Microsoft Typography

[FOT10] http://www.truetype.demon.co.uk/

- TrueType Typography

[FOT11] http://eyegene.ophthy.med.umich.edu/unicode/fontguide/

- List of Open Source fonts.

[FOT12] http://freetype.org/

- FreeType Library: A Free, High-Quality, and Portable Font Engine and TrueType Library which comes with most GNU/Linux Distributions.

[FOT13] http://sourceforge.net/projects/fonttools/

- TTX/Font Tools: TTX is a tool to convert OpenType and TrueType fonts to and from XML. FontTools is a library for manipulating fonts, written in Python. It supports TrueType, OpenType, AFM and to an extent Type 1 and some Mac-specific formats.

[FOT14] http://home.kabelfoon.nl/~slam/fonts/

- TrueType Hinting utilities.

[FOT15] http://savannah.gnu.org/projects/fontutils/

- GNU Font Tools.

[FOT16]http://packages.debian.org/testing/admin/defoma

- Defoma: Font Manager utility for Debian GNU/Linux Distribution.

[FOT17]http://pegasus.rutgers.edu/~elflord/font_howto/ttfutils-0.2.tar.gz

- Ttfutils: TrueType Font Utilities.

[FOT18]http://fontforge.sourceforge.net/overview.html

- FontForge manual.

[ICS1] http://indic-computing.sourceforge.net/handbook/tutorial-issues.html

- This write-up will talk about the issues involved in representing Indian languages on computer. (In progress)

[ICS4] http://indic-computing.sourceforge.net/handbook/tutorial-encodings.html

- Tutorial on encoding (incomplete)

[KNF1] http://www.baraha.com/html_help/baraha/baraha_fonts.htm

- List of Kannada fonts used by Baraha.

[OMT1]
http://indic-computing.sourceforge.net/handbook/tutorial-output-methods.html

- A write-up on output methods (In progress. Just tells what the write-up is going to talk about.)

[ORF1] http://oriya.sarovar.org/download/utkalm.ttf.gz
(http://oriya.sarovar.org/download/utkalm.ttf.gz)

- Open type Oriya fonts

[OTF1] http://store.adobe.com/type/opentype/main.html

- A document on Open Type Fonts

[OTF2] http://partners.adobe.com/public/developer/opentype/index_tag3.html

- List of registerd features of OpenType format.

[OTF3] http://partners.adobe.com/public/developer/opentype/index.html

- OpenType format specification from Adobe.

[MLF1] http://smc.sarovar.org/font-install.html

- This document details the steps for setting up Malayalam fonts and leyboard for GNU/Linux

[RIS1] http://indic-computing.sourceforge.net/handbook/x-indic.html

- Talks about the issues involved in rendering indic scripts in the X Window System (In progress)

[SIF11] http://www.fonts.lk/ (http://www.fonts.lk/)

- Unicode compatible fonts for Sinhala

[TFC1] http://www.baraha.com/html_help/baraha/fontconvert.htm

- Font Conversion utility

21.2. Free Open Source Software Philosophy

[FOSS1]http://www.fsf.org/

- Free Software Foundation

[FOSS2]http://www.opensource.org/

- OpenSource Initiative

[FOSS3]http://www.fsf.org/philosophy/free-sw.html

- Free Software Definition

[FOSS4]http://www.opensource.org/docs/definition_plain.php

- Open Source Definition

[FOSS5]http://www.gnu.org/philosophy/free-software-for-freedom.html

- Free Software Vs Open Source Software

21.3. Input methods and keyboards

[IMH1] http://www.openi18n.org/

- Home page for Internet Intranet Input Method Framework Project.

[IMH2] http://www.freedesktop.org/Software/scim

- Home page for scim project. (scim (Smart Common Input Method) is an XIM Server, which is widely used to input texts in CJK languages.)

[IMS1] http://www.xfree86.org/4.2.0/xmodmap.1.html

- A manual for XMODMAP.

[IMS2] http://www.opencjk.org/~yumj/chinput/ximproto.html

- X Input Method Protocol Specification.

[IMS3] The X KeyBoard Extension Library Specification; Version 11,Release 6.4; Amber J Benson and Gary Aitken, Erik Fortune, Donna Converse, George Sachs, Will Walker .

[IMS4] The X KeyBoard Extension Protocol Specification; Version 11,Release 6.4; Erik Fortune (Silicon Graphics, Inc.)

[IMS5] X Window System Protocol, X Consortium Standard; X Version 11 Release 6; Robert W. Scheifler

[IMS6] The Input Method Protocol, Version 1.0, X Consortium Standard;X Version 11,Release 6.4, Masahiko Narita, Hideki Hiura.

[IMS7]http://www.barcodeman.com/altek/mule/scandoc.php

- Key scan-codes.

[IMW1] http://www.openi18n.org/subgroups/im/iiimf/whitepaper/whitepaper.html

- This document contains whitepaper on Internet/Intranet Input Method Architecture for java, by Hideki Hiura (The original creator of XIM and IIIMF).

[IMW2] http://developer.gnome.org/doc/whitepapers/gtki18n/gtki18n.html (http://developer.gnome.org/doc/whitepapers/gtki18n/gtki18n.html)

- An article on Internationalisation in GTK+.

[IMW3]http://linux.thai.net/%7Ethep/presents/PAN/theppitak-gtk-l10n-slide.pdf

- An article on Internationalisation in GTK+.

[TIM1] http://indic-computing.sourceforge.net/handbook/tutorial-input-methods.html

- Write-up on Input methods

[TIM2] http://tldp.org/HOWTO/Keyboard-and-Console-HOWTO.html

- A tutorial on Linux Keyboard and console.

[TIM3] http://www.linux.org/docs/ldp/howto/Danish-HOWTO-2.html

- A tutorial on Linux keyboard setup.

[TIM4] http://pascal.tsu.ru/en/xkb/

- A tutorial on XKB by Ivan Pascal

[TIM5] http://wauug.erols.com/~balsa/linux/deadkeys/

- An article on Dead Keys.

[TIM6] http://www.jw-stumpel.nl/stestu.html

- An article on multilingual text on Linux.

[TIM7] http://www.suse.de/~mfabian/suse-cjk/suse-cjk.html

- An article on CJK support in SUSE Linux.

[TIM8] http://www.nongnu.org/sinhala/doc/howto/sinhala-howto.html

- An article on Sinhala Support in GNU Linux.

[TIM9] http://web.mit.edu/answers/xwindows/xwindows_xmodmap.html

- An article on XMODMAP, to modify key and mouse actions.

[TIM10] http://indlinux.org/wiki/index.php/IiimfHowto

- A IIIMF How-To on indlinux.org. (Indlinux was one of the first sites to intiate Localisation activities in India).

[TIM11] http://www.indlinux.org/wiki/index.php/KeyboardLayoutSwitching

- This document puts down the procedure involved in changing keyboard layouts in KDE and GNOME.

[TIM12] http://www.indlinux.org/keymap/keymaps.php

- Links to Inscript keymaps for various languages. This will be useful for someone while typing text in various languages.

[TIM13] http://www.suse.de/~mfabian/suse-cjk/xim.html

- A list of existing XIM servers available.

[TIM14] http://www.suse.de/~mfabian/misc/xim

- A useful resource on XIM Server.

[TIM15]
http://docsun.cites.uiuc.edu/sun_docs/C/solaris_9/SUNWalab/SIMPCHNUG/p4.html

- A sun tutorial on htt Input Method Server.

[TIM16]http://developers.sun.com/dev/gadc/technicalpublications/presentations/iuc26-unicode-table-based-input-method-handout-notes.pdf

- UNIT language engine specification.

21.4. KDE Localisation

[KDE1] https://mail.kde.org/mailman/listinfo/kde-i18n-doc/

- Documentation of KDE Internationalisation.

[KDE2] http://i18n.kde.org/teams/index.php

- List of language and coordinators for KDE localisation.

[KDE3] http://i18n.kde.org/stats/gui/

- KDE Internationalisation status table.

[KDE4] http://i18n.kde.org/translation-howto/gui-todo-lists.html

- KDE gui translation to-do list.

[KDE5] http://i18n.kde.org/translation-howto/check-gui.html#check-context

- Semi-Automatic tests for finding the duplicate shortcut keys used in the translations.

[KDE6] http://bugs.kde.org/

- The place where you can submit bugs in the KDE.

[KDE7] http://webcvs.kde.org/kde-i18n/hi/messages/kdebase

- The PO files of KDE for the Hindi language.

[KDE8] http://i18n.kde.org/translation-howto/doc-translation.html

- Document explaning the doc-translation of KDE.

21.5. GNOME Localisation

[GNM1] http://www.gnome.org/

- GNOME website.

[GNM2] http://developer.gnome.org/projects/gtp/teams.html

- List of langauge teams working on GNOME Localisation.

[GNM3] http://developer.gnome.org/projects/gtp/

- GNOME Localisation Website.

[GNM4] http://l10n-status.gnome.org/

- Lanaguage wise status of GNOME Localisation.

[GNM5] http://cvs.gnome.org/viewcvs/

- Online CVS of GNOME.

[GNM6] http://developer.gnome.org/doc/tutorials/gnome-i18n/translator.html

- Document on using GNOME CVS as a translator.

[GNM7] http://developer.gnome.org/projects/gtp/glossary/

- Glossary of the commonly used terms in GNOME and their translations for different languages.

[GNM8] http://cvs.gnome.org/

- GNOME CVS server.

[GNM9] http://developer.gnome.org/projects/gtp/contact.html

- Contact page of GNOME Translation Project team.

21.6. Mozilla Firefox Localisation

[MOZ1] http://www.mozilla.org/

- Mozilla Homepage.

[MOZ2] http://www.mozilla.org/projects/firefox/l10n/

- Mozilla Firefox Loclisation website (maintains the list of localiation teams).

[MOZ3] http://ftp.mozilla.org/pub/mozilla.org/firefox/releases/

- Download area of language packs of various releases of Mozilla Firefox.

[MOZ4] http://www.mozilla.org/projects/l10n/mlp_otherproj.html#ffox_contrib

- Language packs for the languages in which translation work is already started can be found here.

[MOZ5] http://lxr.mozilla.org/aviarybranch/source/toolkit/locales/

- Common files that are to be translated for Mozilla Thunderbird and Firefox are kept here.

[MOZ6] http://lxr.mozilla.org/aviarybranch/source/browser/locales/

- Mozilla Firefox files that are to be translated for Mozilla Firefox are kept here.

[MOZ7] http://lxr.mozilla.org/aviarybranch/source/

- Mozilla CVS Tree.

21.7. OpenOffice Localisation

[OOL1] http://l10n.openoffice.org/

- Localisation Home page on OpenOffice.org

[OOL2] http://www.indlinux.org/wiki/index.php/OpenOfficeLocalization

- Indlinux How-To on OpenOffice Localisation.

[OOL3]
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/intl/nls_238z.asp

- MSLANGID from Microsoft site. (MSLANGID was used in OpenOffice.org version 1.x to identify languages).

[OOL4] http://www.w3.org/WAI/ER/IG/ert/iso639.htm

- ISO 639 Language Codes

[OOL5] http://www.iso.org/iso/en/prods-services/iso3166ma/02iso-3166-code-lists/list-en1.html

- Country name codes.

[OOL6] http://eis.services.openoffice.org/EIS2/servlet/GuestLogon

- Environment Information System to locate and browse OpenOffice.org source from the CVS tree.

[OOL7] http://translate.sourceforge.net/

- Home Page of translate project which aids in the localisation process of Several Softwares including OpenOffice.org.

[OOL8] http://freshmeat.net/projects/openoffice_po/

- Information related to OpenOffice.org Localisation using PO files.

[OOL9] http://ooo.ximian.com/hackers-guide.html

- A hackers guide to OpenOffice.org

[OOL10] http://www.ncb.ernet.in/bharateeyaoo/projecttechmicals.html

- Home page of an OpenOffice.org Localisation project, by CDAC (formerly NCST).

[OOL11] http://books.evc-cit.info/book.php

- OpenOffice.org XML Essentials.

[OOL12] http://l10n.openoffice.org/servlets/ProjectMailingListList

- OpenOffice.org mailing list page

[OOL13] http://openoffice.lugos.si/down/index_en.html

- Page for a Localisation Project for OpenOffice.org in Slovenian.

[OOL14] http://www.khmeros.info/tools/localization_of_openoffice_2.0.html

- A document written by Javier Sola (http://www.khmeros.info/Khmeros_jsola.html) on Localisation of OpenOffice2.0.

[OOL15] http://blog.janik.cz/archives/2004-11-08T21_46_53.html

- A page from Pavel Janik's diary which explains how to localise OpenOffice using translate tools.

[OOL16] http://anoncvs.services.openoffice.org/

- Online CVS of Openoffice.org.

[OOL17] http://tools.openoffice.org/dev_docs/OOo_cws.html

- A document on OpenOffice.org CVS.

[OOL18]
http://tmp.janik.cz/OpenOffice.org/Languages/OpenOffice.org-Languages.html

- List of languages and their codes working on OpenOffice.org localisation.

[OOL19] http://l10n.openoffice.org/textattr.src.out.txt

- Sample output file by generated by transex3.

[OOL20] http://l10n.openoffice.org/L10N_Framework/Intermediate_file_format.html

- L10N Framework, Intermediate file format.

[OOL21]
http://l10n.openoffice.org/L10N_Framework/How_to_localize_and_build_OpenOffice.html

- L10N Framework, rebuild step.

21.8. Glossary

[GLT1] http://hokum.freehomepage.com/Content/Writing/GlossCrte.html

- A write-up on what is a glossary and how-to create a glossary.

[GLT2] http://www.bengalinux.org/wiki/index.php/BanglaGNULinuxGlossary

- This page contains the glossary of related terms used in Bengali Translation (terms are useful for understanding the layout features and script rules)

[GLT3] http://www.mozilla.org/projects/l10n/mlp_glossary.html

- Translation Glossaries in many languages, derived from Mozilla localisation.

[GNG1] http://www.indlinux.org/downloads/gnomegloss/GnomeGlossary.pot

- Gnome glossary - source for much of the technical terms. (In .pot file format!)

[GNG2] http://developer.gnome.org/projects/gtp/glossary/

- This page contains translation glossaries that can be used for translating GNOME application strings.It also gives the steps to update the glossaries, create a new glossary etc.

[TRG1] http://www.indlinux.org/wiki/index.php/HindiGlossary

- Contains suggested translations for some computer terms. (very few terms covered. The link says that it is an extension of the GnomeGlossary.)

21.9. Tools

[KLT1] http://i18n.kde.org/tools/

- List of localisation tools which will be useful in KDE localisation

[TBR1] http://www.baraha.com/html_help/sdk_docs/baraha_direct.htm

- Baraha's BarahaDirect utility - This is an application which can be used to type Kannada/Devanagari text directly into MS Word, PageMaker etc. This page explains how to use the utility.

[TBR2] http://www.baraha.com/html_help/sdk_docs/baraha_sort.htm

- Baraha's Sorting utility : Can be used to sort Kannada words in alphabetical order. (Could not download.)

[TGL1]
http://instcomp.spjc.edu/cfdevelop/alans/webcttemplates/glossarytool/glossarytool.cfm

- Tool to create a simple glossary.

[TLB1] http://lekho.sourceforge.net/

- Lekho is a Bangla Unicode editor. This site gives information on Lekho.

[TLN1] http://www.translate.com/technology/tools/index.html

- Three free-ware tools (on Windows platform) for localisation.

[TLN2] http://www.mozilla.org/projects/l10n/mlp_tools.html

- Tools for localisation including editors, and some tools specific to Mozilla.

[TLN3] http://gtranslator.sourceforge.net/

- Granslator : GNOME based tool for translating PO files.

[TLN4] http://i18n.kde.org/tools/kbabel/

- Kbabel : KDE based tool for translating PO files.

[TLN5] http://poedit.sourceforge.net/

- POedit : A PO translation tool for both GNU/Linux and Microsoft Windows.

[ENC1]http://www.unicode.org/onlinedat/products.html

- List of Unicode enabled products.

[DCT1] http://dictionaries.mozdev.org/installation.html

- List of dictionaries for some European languages which can be downloaded.

[TTR1] http://www.indlinux.org/wiki/index.php/TranslatorsHowto

- Write-up on handling translations for localisation, including information on .po file format, translation tools and Unicode editors.

[TYU1] http://www.indlinux.org/doc/yuditguide.html

- A guide to install and use yudit.

[TYU2] http://www.yudit.org/

- Information on Yudit editor including download, installation, etc.

21.10. Guidelines

[HGL1]http://tldp.org/HOWTO/Bangla-HOWTO//
(http://tldp.org/HOWTO/Bangla-HOWTO/)

- This document explains how to setup and develop support for Bangla (or Bengali) in a GNU/Linux systems.

[HGN1] http://developer.gnome.org/projects/gtp/l10n-guide/

- How-to guide to localise GNOME applications.

[HGN2] http://developer.gnome.org/doc/tutorials/gnome-i18n/developer.html

- GNOME localisation guide - includes how to use gettext and intltool for localisation.

[HLS1] http://www.indlinux.org/wiki/index.php/LanguageHowto

- This document talks in brief about the various steps involved in localising a software to a new language, generally in a community model.

[HLS2] http://www.microsoft.com/technet/archive/ittasks/plan/sysplan/glolocal.mspx

- Write-up on the localisation process includes a case study.

[HLS3] http://www.ncst.ernet.in/projects/indix/e_introduction.shtml

- This document contains technical details that would be useful for localisation.

[TRG2] http://smc.sarovar.org/trans-guide.html

- A document which guides you in translating .po files to Malayalam. Most of the steps detailed here can be used for translation into other languages also.

[TRG3] http://www.indlinux.org/wiki/index.php/TranslatorsHowto

- Write-up on handling translations for localisation, including information on .po file format, translation tools and Unicode editors.

[TRG4] http://developer.gnome.org/projects/gtp/style-guides/

- This page contains links to various style guides given by Sun for translating Sun software.They could be used for any project.

[TRG5] http://developer.gnome.org/documents/style-guide/locale.html (http://developer.gnome.org/documents/style-guide/locale-5.html)

- Guidelines on writing translations (general material)

21.11. System specific pointers

[HGL2] http://tldp.org/HOWTO/Bangla-HOWTO/

- Bangla Linux - How to

[HGN3] http://developer.gnome.org/doc/tutorials/gnome-i18n/translator.html

- For people who are involved in GNOME translation activities, GNOME allows them to apply for a CVS account to do the same. This document explains how to use the cvs account for translation.

[HGN4] http://www.gnome.org/~malcolm/i18n/

- A detailed guide on Internationalising GNOME applications

[HGN5] http://l10n-status.gnome.org/gnome-2.8/index.html and http://l10n-status.gnome.org/gnome-2.6/index.html

- These pages contain statistics on localisation of Gnome-desktop, Office, etc for various languages. It also contains links to the .pot/.po files for all these languages along with current translations available.

  ```
  The po files contain data in the following format

  #: atk/atkobject.c:389
  msgid "Is used to notify that the table summary has changed"
  msgstr "Word gebruik om aan te gee dat die tabelsamenvatting is verander"

  where 'msgid' stands for the actual string and 'msgstr' is the translation
  of the string in the corresponding language.
  ```

[HKD1] http://kannada.sourceforge.net/docbook/t1.html

- A detailed HOWTO on translating KDE and GNOME into kannada. (In progress)

[HKD2] http://i18n.kde.org/translation-howto/

- How-To guide on KDE Translation. This talks about howto bring a new language in KDE, GUI translation , Documentation translation of KDE etc.

[HKD3] http://developer.kde.org/documentation/library/kdeqt/kde3arch/kde-i18n-howto.html/

- This document talks about how to prepare KDE applications for localisation.Also contains to some tools

[HKD4] http://i18n.kde.org/stats/doc/

- Detailed status of KDE documentation message translations

[HKR1] http://www.ncst.ernet.in/projects/indix/HOWTO/DevaHowto/Devanagari-HOWTO-index.html

- Changing X-kernel for localisation - the Indix approach.

[HKR2] http://lwn.net/Articles/28916/

- An article on Kernel Internationalisation.

[HKR3] http://lwn.net/Articles/28916/

- An article on Kernel Internationalisation.

[HKR4] http://mail.nl.linux.org/linux-utf8/2005-01/#00020

- A discusson thread on Console localisation.

[HKR5] http://mail.nl.linux.org/linux-utf8/2005-01/msg00061.html

- A discusson thread on Console localisation.

[HMZ2] http://www.mozilla.org/projects/l10n/mlp_howto.html

- Details the necessary steps and available resources to create a localised version of mozilla - very rich page.

[HOO1] http://kannada.sourceforge.net/openoffice.org/

- A useful link for openoffice localisation in Kannada

[HOO2] http://l10n.openoffice.org/#l10n

- Openoffice localisation - guidelines, glossary, etc

[HOO3] http://l10n.openoffice.org/i18n_framework/HowToAddEncoding.html

- A write-up on howto add a new text encoding to OpenOffice.org (*)

[HOO4]
http://l10n.openoffice.org/L10N_Framework/How_to_localize_and_build_OpenOffice.html

- This document explains in detail the steps required to localise and build OpenOffice 1.x.

[HOO5] http://l10n.openoffice.org/L10N_Framework/iso_code_build2.html

- This document explains in detail the steps required to localise and build OpenOffice 2.x

[IGT1] http://developer.gnome.org/doc/whitepapers/gtki18n/index.html

- A document that describes the internationalisation process for GTK+

21.12. Localisation Projects

[DLN1] http://www.debian.org/intl/l10n/

- Debian localisation.

[FLN1] http://fedora.redhat.com/projects/translations/

- Localisation of Fedora core.

[LBA1] http://www.bengalinux.org/projects/distro/ and
http://sourceforge.net/project/showfiles.php?group_id=43331/
(http://sourceforge.net/project/showfiles.php?group_id=43331)

- These pages contain a list of localised Bangla Application and resources

[LPB1] http://www.baraha.com/

- This site gives information about Baraha which is a word processing applications for creating documents in Indian languages. It also has links to download some utilities

[LPB2] http://www.baraha.com/baraha50.htm

- You can download BarahaDirect from here.

[LPR1] http://www.iosn.net/l10n/projects/

- This page contains links to various localisation projects going on in East Asia, South-East Asia, South Asia and the Middle-East.

[LPU1] http://utkarsh.org/?q=downloads

- This page contains some downloads including Gujarati fonts , Utkarsh installer (Utkarsh is Linux in Gujarati)

[TPR1] http://www.iro.umontreal.ca/translation/HTML/

- This is a Traslation Project

21.13. Unclassified!

[EIS1] http://www.m17n.org/emacs-indian/

- Indian script support in Emacs

[HGL3] http://oriya.sarovar.org/docs/getting_started_single.html

- How to Linux in Oriya

[HGL4] http://www.tamillinux.org/?q=book/view/12

- Tamil Linux How to

[ILO1] http://www-306.ibm.com/software/globalization/topics/indic/index.jsp

- An overview of Indic languages.

[ITL1] http://www.indictrans.org/

- A site with different kinds of information including write-ups on fonts,links to translation aids,write-ups on input methods etc.

[MSL1] A Reference Model For Software Localisation (S.P.Mudur, Rekha Sharma)

[PNL1] http://punlinux.sourceforge.net/

- Punjabi linux

[ROL1] http://oriya.sarovar.org/index.html (http://oriya.sarovar.org/index.html)

- Rebati - Oriya localisation project

[TCV1] https://www.cvshome.org/new_users.html

- A document on what CVS is. This document also contains links to further readings on CVS.

[TCV2] https://www.cvshome.org/docs/

- An extensive material on CVS. Includes what is CVS, howto use CVS and frequently asked questions.

[TCV3] https://www.cvshome.org/docs/blandy.html

- Good write-up on CVS.

[TLC1] http://indic-computing.sourceforge.net/handbook/tutorial-locales.html

- This article will contain a discussion on locales. (In progress)

[TLC2] http://www.opengroup.org/onlinepubs/007908799/xbd/locale.html

- A write-up on locale: how to create, the various components and their options.

[TLC3] http://l10n.openoffice.org/i18n_framework/HowToAddLocaleInI18n.html

- This document talks about how to add a new locale to the i18n framework. (*)

[TLC4] http://www.loc.gov/standards/iso639-2/englangn.html

- 3-letter language code as defined in ISO 639-2

[TLC5] http://www.iso.org/iso/en/prods-services/iso3166ma/02iso-3166-code-lists/list-en1.html

- 2-letter country code as defined in ISO 3166

[TLC6] http://sources.redhat.com/ml/libc-locales/2005-q1/msg00002.html

- A discussion on Locale submission and maintenance.

[TLC7] http://www.student.uit.no/~pere/linux/glibc/

- GNU libc Patch page.

[TLC8] http://sources.redhat.com/bugzilla/

- Redhat's Bugzilla

[TLC9] http://sources.redhat.com/bugzilla/enter_bug.cgi

- Interface for submitting a locale

[TLC10] http://publib.boulder.ibm.com/infocenter/pseries/index.jsp?topic=/com.ibm.aix.doc/files/aixfiles/LC[/LC_TIME.htm]

- LC_TIME : explantion on Field descriptors.

[TLC11]
http://publib.boulder.ibm.com/infocenter/pseries/index.jsp?topic=/com.ibm.aix.doc/files/aixfiles/LC
[/LC_COLLATE.htm]

- Collation-symbol: Keywords explanation.

[TLC12]
http://publib.boulder.ibm.com/infocenter/pseries/index.jsp?topic=/com.ibm.aix.doc/files/aixfiles/LC
[/LC_MONETARY.htm]

- LC_MONETARY: Keywords explanation.

[TRG7] http://www.completetranslation.com/principles.htm

- A good document on Principles of Translation.

[TRG8] http://kannada.sourceforge.net/openoffice.org/files/kn-gsi-26-may-2004.po.gz

- .po files containing Kannada translations. Will be a useful reference for someone translating an application into Kannada.

[XIL1] http://www.opentag.com/xmli18nfaq.htm

- Answers to some of the most frequently asked questions about XML internationalisation and localisation

[POT1]http://www.gnu.org/software/gettext/manual/html_node/gettext_9.html#SEC9

- Gettext manual and PO file format.

[PLH1]http://www.dwheeler.com/program-library/Program-Library-HOWTO/

- Creation of Program Libraries- HOWTO.

[GTM1]http://www.gnu.org/software/gettext/manual/html_mono/gettext.html

- GNU Gettext Manual.

[GTM2]http://www.gnu.org/software/gettext/manual/html_mono/gettext.html#SEC172

- GNU Gettext Utilities.

[GTM3]http://www.gnu.org/software/autoconf/manual/autoconf-2.57/html_mono/autoconf.html

- Autoconf

[WZP1] http://www.winzip.com/

- Winzip Homepage.

Glossary

Anchor

: anchor points are used in font editors like Fontforge to attach two glyphs.

  See Also: Glyph.

Anti-aliasing

: A technique for providing smoother looking characters. Scaling of characters results in sharp, jagged edges. Anti-aliasing is the process of smoothening the jagged edges

  See Also: Hinting.

Ascender

: Ascender denotes that portion of a character that extend above the baseline (e.g. Uppercase English characters).

See Also: Ascender Space, Baseline, Character Cell, Descender.

Ascender Space

Ascender Space denotes that portion of character which appears above the base character

See Also: Ascender, Baseline, Character Cell, Descender.

aspell

A spell checker library/utility

See Also: ispell, Spell.

Baraha

A multilingual text processing tool (for Windows). The transliterated texts in Roman scripts are converted to the other scripts it supports.

See Also: Emacs, Mozilla Composer, Simredo, Yudit.

Baseline

Baseline is an imaginary line on which majority of characters rest.

See Also: Ascender, Ascender Space, Character Cell, Descender.

BharateeyaOO.o

A project on localisation of OpenOffice.org in Indian Languages, by CDAC (formerly NCST) .

See Also: OpenOffice.org.

Bitmap Font

A font whose character shapes are defined by arrays of bits.

See Also: Font, Outline Font, Vector Font.

Bugzilla

Bugzilla is the world-leading free software bug tracking system, which is now industry-standard bug tracking system. It is quite popular in Open Source community.

catgets

catgets is a system's message catalog tool/library. It is used to retrieve messages from a message Catalog. The catgets implementation is defined in the X/Open Portability Guide.

(Note:The Open Source Community is divide into the use of two message catalog specification namely catgets and gettext)

See Also: Message Catalog, gettext.

Character Cell

Character cell is a square area which is assigned to a character on a computer screen.

See Also: Ascender, Ascender Space, Baseline, Descender.

Character Encoding

A table which maps a character code to a glyph.

See Also: Single Byte Encoding, Multi Byte Encoding.

Character Set

A set of numeric codes used by a computer system to represent a standard collection of characters. These characters may be local to particular country or place.

See Also: Font.

Collation

Lexicographical/alphabetical ordering of textual information. Collation is defined relative to a single language. Collation order for Unicode characters is defined in Unicode Technical Report #10, "Unicode Collation Algorithm," (http://www.unicode.org/reports/tr10/)

Community Model

A Software development Model in which there is an active participation of the community, and community gives direction to the development according to its requirements. This is also known as Open Model.

See Also: Non-Community Model.

Concurrent Versions System

A code management system, which has the ability to track incremental changes (made by different people concurrently) to files.These changes can also be reverted

back. CVS is used by a number of software development groups to keep track of their projects.

convmv

A filename encoding converter.

See Also: iconv, International Components for Unicode, uconv, uniconv.

Cultural Localisation

Adapting the software to the cultural conventions of the target user community

See Also: Localisation, Internationalisation.

Descender

Descender denotes that portion of a character which extends below the baseline (e.g. portions of English letters like 'g', 'j', 'p', 'q', 'y').

See Also: Ascender, Ascender Space, Baseline, Character Cell.

Devanagari

A syllabic script used in writing languages like Hindi, Sanskrit, Marathi, Nepali etc.

Doc Translation

Some reasonable definition here.

Emacs

A general purpose editor which can be used to edit PO files.

See Also: Baraha, Mozilla Composer, Simredo, Yudit.

Font

A visual design for a character set.

See Also: Character Set, Font Designer, Font Metric, Font Shape, Typeface, Typography, Bitmap Font, Outline Font, Vector Font.

Fontconfig

A generic Font configuration library designed to provide system-wide font configuration and customisation.

See Also: Font.

Font Designer

One who designs a Font. Font design may deal with issues like character size, character dimension and other calligraphic informations.

See Also: Font.

Fontforge

A tool for creating Fonts

See Also: Font, gfonted, ttf2bdf, xmbdf.

Font Metric

Font metrics define the spacing between variable width fonts. It include information like size of the font, kerning information etc.

See Also: Font, Font Shape, Typeface, Typography.

Font Shape

This is the outline or shape of the font. The components of font shape is a stroke, an accent etc. The font shape is defined in terms of the glyphs for that particular character set.

See Also: Font, Font Metric, Typeface, Typography.

FOSS

An abbreviation for Free and Open Source Software.

See Also: Free Software, Open Source.

Free Software

A general term which is used to describe a software which is available freely for modification and redistribution.

See Also: Open Source, Free Software Foundation.

Free Software Foundation

A non-profit organisation, founded by Richard M. Stallman, dedicated to the philosophy of production and (re)distribution of Free Software.

See Also: Free Software, GNU.

FriBidi

A free implementation of Unicode Bi-Directional Algorithm. Internally the library uses Unicode entirely.

See Also: protobidi.

Fuzzy

A word used to mark those strings which have ambiguous or indistinct translation..

See Also: Translation Memory.

Gedit

A lightweight text editor for GNOME desktop environment .

gettext

gettext is the most popular message catalog tool/library. It is a multilingual GNU Internationalisation tool/library (contained in libintl library). It is mostly used, with PO files, to Localise an application.

(Note:The Open Source Community is divide into the use of two message catalog specification namely catgets and gettext)

See Also: Message Catalog, catgets, xgettext, msgfmt, msgconv, msgmerge, Portable Object file, Machine Object file, libintl.

gfonted

A font editor for Adobe Type 1 format fonts. It is currently in the early stages of development.

See Also: Font, Fontforge, ttf2bdf, xmbdf.

Globalisation

A product development approach which enables the worldwide market accessible for a software product, through the extensive use of Internationalisation and Localisation.

See Also: Internationalisation, Localisation.

Glossary

In the context of localisation, a glossary is a list of English terms that are used to define software terms and their translations in a foreign language.

Glyph

A visual representation of a character. A character can have a number of glyphs.

See Also: Ligature.

Glyph Positioning Table

The GPOS table stores "positioning information" which can be used to position the glyphs accordingly or relative to each other based on the GPOS data.

See Also: Glyph Substitution Table, Glyph, Ligature.

Glyph Substitution Table

The GSUB table stores "substitution information" which can be used to substitute a glyph/glyphs (or ligature/ligatures) with different glyph/glyphs (or ligature/ligatures).

See Also: Glyph Positioning Table, Glyph, Ligature.

GNOME

An abbreviation for GNU Network Object Model Environment. GNOME is a very popular Window Manager/Desktop Environment for GNU/LINUX.

See Also: KDE.

GNU

An abbreviation for GNU's not UNIX. It is a project started by Free Software Foundation to support freely distributable POSIX compliant softwares.

See Also: Free Software Foundation.

GTK+ IM

An Input Method framework for GTK+ applications.

See Also: Input Method, Internet Intranet Input Method Framework, X Input Method, X Keyboard Extension.

gtranslator

A GUI IDE/editor (on GNOME) for gettext PO files.

See Also: KBabel, poedit.

GUI Translation

A term for Translation using GUI tools. GUI tools make the translation process considerably easier.

Hinting

A hint is a mathematical instruction added to the font. These instructions are used while displaying the characters. Hinting decides what pixels to be turned on while displaying the character so that we will get the best possible bitmap image. Hinting is important at smaller resolutions as less number of pixels are available to display a character.

See Also: Anti-aliasing.

iconv

iconv converts the encoding of characters in a file to a different encoding. This utility can also be used to convert data (in form of text) from legacy applications,in old encoding formats, to Unicode or UTF-8 encoding formats.

See Also: convmv, International Components for Unicode, uconv, uniconv.

ICT

abbreviation for Information and Communication Technology.

Input Method

A facility for entering any ASCII/Non-ASCII characters set using a standard keyboard.

See Also: Internet Intranet Input Method Framework, GTK+ IM, X Input Method, X Keyboard Extension.

International Components for Unicode

An utility with Unicode support. It is a Unicode framework which has a set of tools/utilities, libraries and API's which help in the conversion of encodings on most of the systems.

See Also: convmv, iconv, uconv, uniconv.

Internationalisation

The process of customising/adapting a software/product in such a way that it can be extendable/adaptable to any local Language, local custom, local culture, local conventions etc, very easily.

See Also: Globalisation, Localisation.

ISO

An abbreviation for International Organisation for Standardisation.

Internet Intranet Input Method Framework

IIIMF is a platform independent, distribution independent, distributed Input Method Framework, which provides Input Method services to its clients..

See Also: Input Method, Internet Intranet Input Method Framework, GTK+ IM, X Input Method, X Keyboard Extension.

ispell

A spell checker tool

See Also: aspell, Spell.

KBabel

An IDE for translation, using PO files.

See Also: gtranslator, poedit.

KDE

An abbreviation for K Desktop Environment. It is a popular desktop environments for GNU/Linux.

See Also: GNOME.

Kernel

The core part of an Operating System that remains in the main memory and interacts and manages the hardware resources.

Kerning

Process of moving two adjacent characters relative to each other (closer or apart). This is done in order to improve the readability of text, space utilisation on page and produce best fit between the letters..

Language

A mode of communication for a community or region. When talking with respect to Computer Font technology, it represents a unique character set for any script.

See Also: Script.

libintl

GNU internationalisation library.

See Also: gettext, xgettext.

Ligature

A single typographic character/glyph, which is formed when two or more characters/glyphs (atomic or non-atomic characters/glyphs) combine together. The Ligature may have a irrelevant shape as compared to its constituent character/glyph components.

See Also: Glyph.

Linguist

A person skilled in Language(s)

LISA

abbreviation for Localization Industry Standards Association

Locale

A term used to define a set of information corresponding to a given language and country (or region). A locale information is used by a software application (or operating system) to exhibit a localised behaviour for that application (if that localised behaviour is defined).

See Also: localedef.

localedef

A utility which is used to define locale environment, by compiling a locale definition file.

See Also: Locale.

Localisation

Localisation is the process of adapting, translating and customising a product for a specific market. The Localisation Industry Standards Association (LISA), defines l10n as "Localisation involves taking a product and making it linguistically and culturally appropriate to the target locale where it will be used and sold."

See Also: Internationalisation, Globalisation.

Localisation Engineer

Responsible for Localisation Engineering of a Software product. They study the feasibility of localising a Software Application, create and provide various technologies for translation process.

See Also: Localisation, Localisation Manager, Localisation Vendor.

Localisation Manager

Manges the overall localisation and translation process by interfacing with various persons/tools responsible. They can be either a Person, Software Entity or Abstract.

See Also: Localisation, Localisation Engineer, Localisation Vendor.

Localisation Vendor

Person/Organisation providing the localisation solution for a Software Product/Application.

See Also: Localisation, Localisation Engineer, Localisation Manager.

Message Catalog

A file/database system (either binary or non-binary), implemented either in a directory hierarchy or as a stand-alone database/file, having a list(s) of strings/messages, which can be retrieved by Software Application(s) (at runt time). Message Catalog (translated Message Catalog) s generally used by localised Software Application(s), to retrieve mesasage(s) in local language(s) (in local character set of the language). There can be either system-wide Message Catalog or application specific Message Catalog.

See Also: gettext, catgets, xgettext, Portable Object file, Machine Object file.

Machine Object file

MO file are binary Message Catalog files used mostly under gettext framework.

See Also: Message Catalog, gettext, xgettext, msgfmt, Portable Object file, Portable Object Template file.

moz2po

This tool converts Mozilla specific xml files to PO files.

See Also: po2moz, Portable Object file.

Mozilla

A popular web browser.

See Also: Mozilla Firefox.

Mozilla Composer

An HTML editor which can be used to edit in Multiple Languages.

See Also: Baraha, Emacs, Simredo, Yudit.

Mozilla Firefox

A popular web browser.

See Also: Mozilla.

Mozilla Translator

A translation tool for Mozilla.

See Also: Mozilla, Mozilla Firefox.

msgfmt

A gettext command, which is used to compile binary Message Catalog. This binary Message Catalog (MO file) is compiled from the non-binary Message Catalog (PO file). This command can also be used to test the validity of the translation in PO file.

See Also: gettext, xgettext, msgconv, msgmerge, Portable Object file, Portable Object Template file, Machine Object file.

msgconv

A gettext command, which is used to convert character encoding of a translated Message Catalog (PO file) to a different encoding.

See Also: gettext, xgettext, msgfmt, msgmerge, Portable Object file, Portable Object Template file, Machine Object file.

msgmerge

A gettext command, which is used to merge strings in Message Catalog (PO file) and Template file (POT file) .

See Also: gettext, xgettext, msgfmt, Portable Object file, Portable Object Template file, Machine Object file.

Multi Byte Encoding

An encoding scheme which uses multiple bytes to represent a character.

See Also: Character Encoding, Single Byte Encoding.

Non-Community Model

An Open Source development Model which does allow the participation of community. It is also known as closed Model.

See Also: Community Model.

Normalization

To convert the data into a form that can be compared for equivalence in a binary format. Normalization is defined in Unicode Technical Report #15, "Unicode Normalization Forms" (http://www.unicode.org/reports/tr15/)

oo2po

This tool converts OpenOffice.org specific GSI/SDF files to PO files.

See Also: OpenOffice.org, Portable Object file.

OpenOffice.org

A free office productivity suite which comes with a collection of office applications. This product is compatible to all other major office software products

See Also: BharateeyaOO.o.

Open Source

A term used to define a software whose source code is freely available for lookup and modification.

See Also: Free Software, Open Source.

Open Source Community

A community or a group of people, who believe in Open Source and Open Source methodologies.

See Also: Open Source.

OpenType Font

A type of font format, developed jointly by Microsoft Inc. and Adobe System Inc., which unifies TrueType and Type1 font format specifications.

See Also: TrueType Font, Type1 Font.

Outline Font

A Font which is defined as a outline of lines and curves. These outlines are define mathematically, allowing it to be rendered smoothly at any size, using mathematical scaling factors.

See Also: Font, Bitmap Font, Vector Font.

Pixel

Pixel is an abbreviation for picture element. It represents the smallest element in a visual display.

po2moz

This tool converts PO files back to Mozilla specific xml files.

See Also: moz2po, Portable Object file.

po2oo

This tool converts PO files back to the OpenOffice.org specific GSI/SDF files.

See Also: OpenOffice.org, Portable Object file.

poedit

An IDE/Editor for editing PO files.

See Also: gtranslator, KBabel.

Portable Object file

A Message Catalog (non-binary) file which is used to provide translation in a particular language. Generally PO files are used under gettext framework to translate the messages of an application. The original messages from the application is identified by the keyword msgid and the translated messages (in your native language) are identified by the keyword msgstr.

See Also: Message Catalog, Portable Object Template file, Machine Object file.

PO Mode

An editing mode for multilingual PO file editors, which eases the editing of the PO/POT files,and takes care that the format of the PO file is not violated.

See Also: Portable Object Template file, Portable Object file.

Portable Object Template file

A POT file is similar to a PO file except that it does not contain any translations and acts as an template file, which contains all the updated messages from the application, which do have any translation towards any language..

See Also: Portable Object file, Machine Object file.

Proof-reader

A person who checks a manuscript to verify the errors in that document, before it is sent for printing.

protobidi

It is a prototype implementation of Bi-Directional Widgets,that has been implemented on top of a graphics library called "cnv". The application uses FriBidi.

See Also: FriBidi.

QA Specialist

One who specialises in Quality Assurance.

Sans Serif

A character set without Serif.

See Also: Serif.

Script

Written form of a language.

See Also: Language.

Serif

Small decorative strokes, stemming from terminating ends of a character, in order to improve the readability of the character.

See Also: Sans Serif.

Simredo

A Jave based multilingual Unicode Editor

See Also: Baraha, Emacs, Mozilla Composer, Yudit.

Single Byte Encoding

An encoding scheme which uses a single byte to represent a character.

See Also: Character Encoding, Multi Byte Encoding.

Spell

A spell checker tool

See Also: aspell, ispell.

Testing Engineer

One who carries out Testing of Softwares.

translate toolkit

translate toolkit comes with a set of toolkit which help in the localisation process.

See Also: oo2po, po2oo, moz2po, po2moz.

Translation

The process of transforming a written text in a language into a written text in another language, with semantics of text being same in both the languages.

See Also: Translator, Transliteration.

Translation Memory

A file/buffer area which stores previously translated texts/strings/phrases. Translation Memory is used for automatic translation of strings/phrases, that have occurred previously in the translation project.

See Also: Fuzzy.

Translator

A person/machine who carries out Translation.

See Also: Translation.

Transliteration

Write the text of one language using the script of another language.

See Also: Translation.

TrueType Font

A type of Font developed by Microsoft Inc.

See Also: OpenType Font, Type1 Font.

ttf2bdf

This tool can be used to convert fonts in TrueType format to fonts in BDF format.

See Also: Font, Fontforge, gfonted, xmbdf.

Type1 Font

A type of font format developed by Adobe Inc. .

See Also: OpenType Font, TrueType Font.

Typeface

Typeface is the style or design of the font. For example bold, italic, and bold-italic are examples of typefaces.

See Also: Font, Font Metric, Font Shape, Typography.

Typography

The look and feel, order and organisation and style of type and typefaces.

See Also: Font, Font Metric, Font Shape, Typeface.

uconv

A character encoding converter and transliteration tool. An utility under IBM's ICU framework.

See Also: convmv, iconv, International Components for Unicode, uniconv.

Unicode

A character encoding standard developed by the Unicode Consortium. Character are encoded in units of length 8 bits(UTF-8), or of length 16 bits(UTF-16) or of length 32 bits(UTF-16).

UNIcode Table based Input Method

UNIT is a generic multilingual language engine, developed under IIIM Framework, for those languages which can be input using generic table lookup approach (e.g. All Indic Languages).

See Also: Internet Intranet Input Method Framework.

uniconv

A character encoding converter. It converts the data from one encoding to another.

See Also: convmv, iconv, International Components for Unicode, uconv.

Vector Font

A scalable font made of vectors

See Also: Font, Bitmap Font, Outline Font.

Web Designer

A person who designs web sites.

Web Manager

A person who manages web sites.

Web-site Builder

A person who builds web sites.

X

Generally used as an abbreviation for X Server or X Protocol.

X Font Server

A background process which makes fonts available to the X Server.

See Also: X FreeType interface.

X FreeType interface

X FreeType interface library provides a client side font API for the X applications.

See Also: X Font Server.

xgettext

A gettext command, which extracts the gettext strings, in an application, to an output file. The out file is generally a PO file.

See Also: Message Catalog, msgfmt, msgconv, msgmerge, Portable Object file, Machine Object file, libintl.

X Input Method

An Input Method Framework which acts as an event (ie. keyboard event, mouse event etc.) filter for X Client.

See Also: Input Method, Internet Intranet Input Method Framework, GTK+ IM, X Keyboard Extension.

X Keyboard Extension

An utility for creating custom keyboard layouts for X11.

See Also: Input Method, Internet Intranet Input Method Framework, GTK+ IM, X Input Method.

Xfree86

A freely (re)distributable Open Source implementation of X Window System (X Consortium)

xmbdf

xmbdf is a bdf (bitmap distribution format) font creation tool.

See Also: Font, Fontforge, gfonted, ttf2bdf.

Yudit

A multilingual text editor. It has its own rendering Engine, can be used to enter transliterated text in Roman script,which are mapped to the desired/selected language/script.

See Also: Baraha, Emacs, Mozilla Composer, Simredo.

Appendix A. Credits/Document History

The need for this guide was felt because it was noticed that there was a lack of knowledge on 'how-to' localise FOSS applications. It was found that even though there were a lot of related resources on the web they were all dispersed and there was no single resource which explained the complete process from start to end. As a result it took an individual or a team with a thorough understanding of FOSS system a substantial amount of time to locate relevant resources, a big hindrance for anyone to take up a localisation project.

This guide is created by a team from C-DAC Mumbai, comprising of Sasikumar M., Aparna R., Naveen K. and Rajendraprasad M., in a time span of about 6 months. Before starting work on this guide the team members tried their hand on localisation by taking some application and localising it. The applications attempted during this phase were Vasistha, an instruction delivery tool developed by C-DAC Mumbai, Mozilla and Moodle. This gave the members an insight into the localisation process. Along side various resources from the web were collected. When a little over hundred resources were collected, a survey of people actively working in localisation was conducted, in order to understand what would be the expectation of a layman from this guide. Useful inputs were received from: Mr. Anousak, Dr. Sarmad Hussain, Mr. Simos Xenitellis, Mr. Javier Sola, Mr. Theppitak Karoonboonyanan and Mr. Shahid Akhtar. All these people were sent an initial sketch of the guide on which they gave their feedback. Based on the feedbacks received, the outline of the guide was drafted and the major chapters and tools to be included were defined. Once this was done work started on individual chapters. When more than 50% of the guide was ready, a Review workshop was organised, on the 7th and 8th of October 2004, to review the work done. The attendees of the workshop included the C-DAC and IOSN team and members from the open source and localisation community: Mr. Abhijit Dutta from IBM, Aman Preet from Punjabi team, Mr. Gora Mohanty from Oriya team, Mr. Javier Sola from Khmer localisation team, Ms. Jayaradha from Tamil localisation team, Mr. Mahesh T. Pai from Malayalam team, Dr. Nagarjuna from FSF India, Mr. Rajesh Ranjan from Hindi team, Mr. Sandeep Rao from Indix team of C-DAC. Individual chapters were reviewed during the workshop and changes necessary were discussed.

Based on the workshop comments, the guide was further revised. A guide of this nature will continue to evolve. Any suggestions are welcomed and can be mailed to info@apdip.net .

Also available from UNDP's Asia-Pacific Development Information Programme

e-Primers on Free/Open Source Software

1. Free/Open Source Software–A General Introduction
2. Free/Open Source Software–Education
3. Free/Open Source Software–Localisation

www.iosn.net

e-Primers for the Information Economy, Society and Polity:

1. The Information Age
2. Nets, Webs and the Information Infrastructure
3. e-Commerce and e-Business
4. Legal and Regulatory Issues in the Information Economy
5. e-Government
6. ICT in Education
7. Genes, Technology and Policy
8. Information and Communication Technologies for Poverty Alleviation

www.eprimers.org

IDRC CRDI
Canada

International Open Source Network

An Initiative of the UNDP's Asia-Pacific Development Information Programme and supported by the International Development Research Centre, Canada

Asia-Pacific Development Information Programme

www.apdip.net

United Nations Service Building
3rd Floor, Rajdamnern Nok Avenue,
Bangkok 10200 ,
Thailand

Tel: +66 (0) 2288 1234, (0) 2288 2129
Fax: +66 (0) 2288 3032