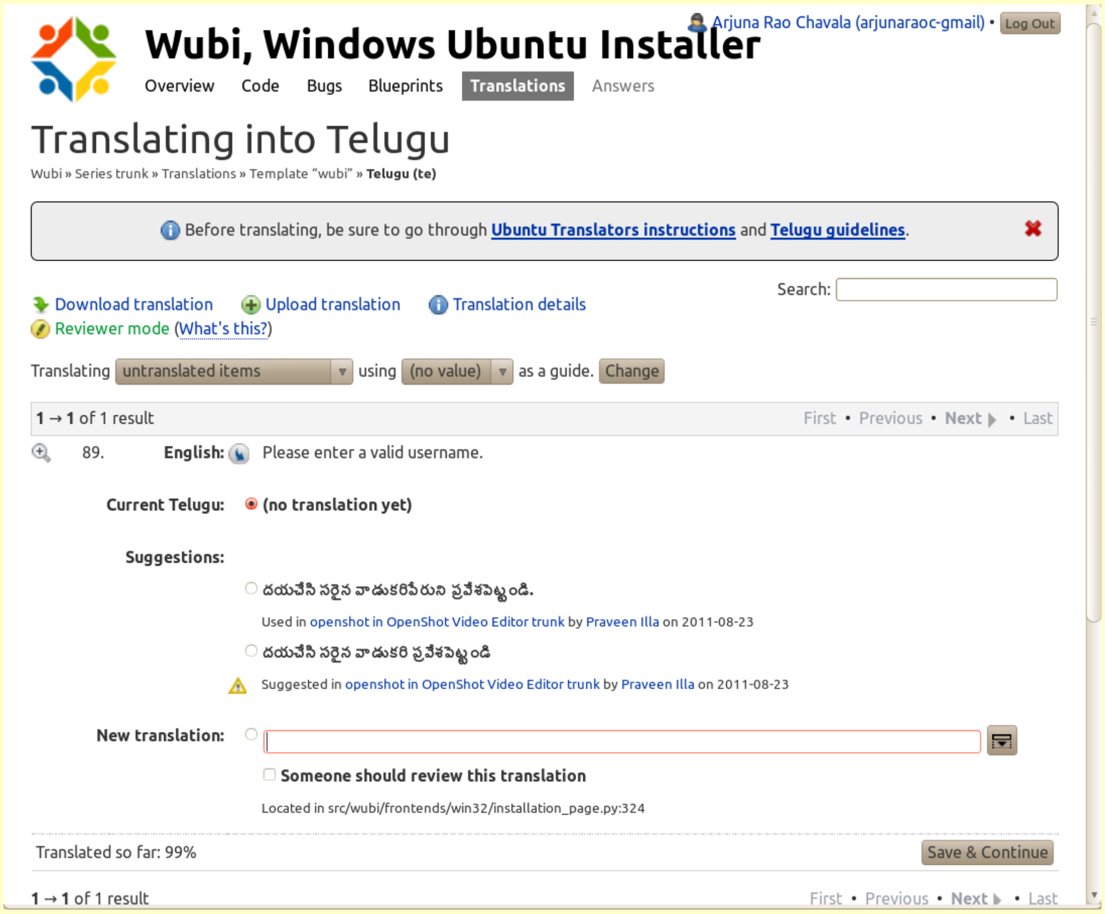
Localisation



Arjuna Rao Chavala (arjunaraoc-gmail) • Log Out

Wubi, Windows Ubuntu Installer

Overview | Code | Bugs | Blueprints | Translations | Answers

Translating into Telugu

Wubi » Series trunk » Translations » Template "wubi" » **Telugu (te)**

Before translating, be sure to go through **Ubuntu Translators instructions** and **Telugu guidelines**.

Download translation | Upload translation | Translation details

Search:

Reviewer mode (What's this?)

Translating untranslated items using (no value) as a guide. Change

1 → 1 of 1 result — First • Previous • **Next** • Last

89. **English:** Please enter a valid username.

Current Telugu: ● **(no translation yet)**

Suggestions:

○ దయచేసి సరైన వాడుకరిపేరుని ప్రవేశపెట్టండి.

Used in openshot in OpenShot Video Editor trunk by Praveen Illa on 2011-08-23

○ దయచేసి సరైన వాడుకరి ప్రవేశపెట్టండి

Suggested in openshot in OpenShot Video Editor trunk by Praveen Illa on 2011-08-23

New translation: ○

☐ **Someone should review this translation**

Located in src/wubi/frontends/win32/installation_page.py:324

Translated so far: 99% — Save & Continue

1 → 1 of 1 result — First • Previous • **Next** • Last

Arjuna Rao Chavala

Chief Consultant, Arc Alternatives

arjunaraoc@arcalter.com

Title: **FOSS Localisation**
Author: **Arjuna Rao Chavala**
Ebook 1st Edition: **Apr 2013**
Copyright: **Arjuna Rao Chavala**
License: **CC-BY-SA 3.0**
Acknowledgements: Linux For You (The individual articles appeared in the magazine from April 2012 to Mar 2013)
Front Cover: Ubuntu 11.04 Telugu, Inside Back Cover: Globalisation Process

Foreword

I started my entry into Free and open source software by attempting to install Redhat operating system for setting up internal email for a small startup software team that I was heading in the year 2000. We used it for browsing websites and email. Subsequently I bought a home computer and installed Redhat. I followed the developments in FOSS and continued to upgrade my system with the latest releases. Around 2005, the software started supporting Indic languages, my interest to use the computer in my mothertongue made me explore a lot more about Indic input and localisation. Seeing that the browser and Openoffice were not available or good quality in Telugu, I worked with net friendst to address these issues. I contributed to the release of Telugu version of Firefox 3.0.2, Libreoffice 3.x and Ubuntu 11.04 over the years.

When I noticed few bugs with Telugu rendering, I attempted to fix them spending huge amount of time exploring different pieces of software. At last I was able to fix them. As the best way to popularise the indic language computing is to get the children to use it as they learn about computers, I made few attempts to meet with School education officials in Andhra Pradesh to impress on them the need for using Telugu for Telugu Medium Students. The response I got was that Students are learning a bit of English through Computer Classes and it is not desirable to deprive them of this opportunity. State like Kerala have started using FOSS in its schools and I thought I can convince them on the merits of that experience. I realised that there is a need for more followup. Being a resident of Bangalore, I am unable to do so. I thought of sharing my learnings and the result is a set of articles published in LFY/OSFY, which I have now compiled to bring out as an e-book.

Bangalore
Date:16 April 2013 Arjuna Rao Chavala

Table of Contents

Part 1: Localisation: An Introduction ...3

Part 2: The Status of Indic Localisation ...5

Part 3:The Gettext framework...8

Part 4: Localise an Application Menu ...11

Part 5:Localise Application Help Files ...14

Part 6:Web-based Platforms for Localisation ...16

Part 7: Localising User Documentation ...19

Part 8: Ensuring consistency in localisation through Style guides ...21

Part 9: The status of Education and Training ...23

Part 10: The status of Research in Localisation ...25

LOCALISATION AN INTRODUCTION

This article is aimed at language computing enthusiasts interested in using computing devices in their native language, developers interested in supporting localisation, linguists interested in understanding translation issues, and business managers exploring the potential of localisation.

If you have ever tried to install an operating system, you will be greeted with several questions concerning the configuration of the system. It could be the location, which determines properties like the time zone, date format and the type of keyboard, and the support for additional languages. These settings also aid in the setting-up of software servers for updates. The technology behind this is called *Localisation*. It is also referred to as *l10n*, as there are ten letters between 'l' and 'n'. For example, Ubuntu 11.04 was released with support for 60 languages.

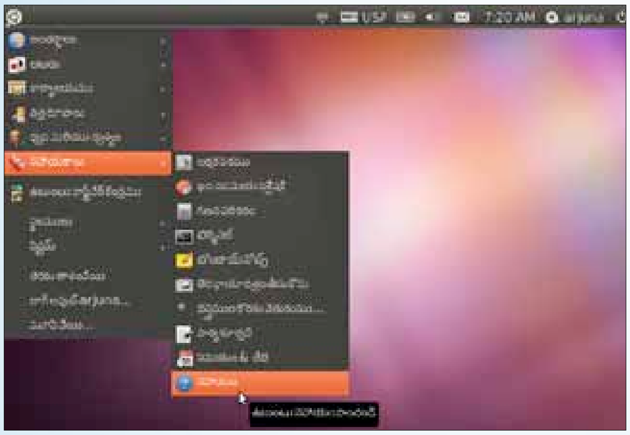


Figure 1: Ubuntu 11.04 in Telugu (http://commons.wikimedia.org/wiki/File:Ubuntu11.04Te2DUnity)

The configuration of a computing device to enable usage in a language chosen by users, and in an environment familiar to them, is called *Localisation*. When computers were originally developed in the West, they only had support for English-speaking countries. As the use of computers increased worldwide, the need to operate the computer in a language of the user's choice grew in importance. Suppliers had to adapt to this need by developing different versions of software and hardware. Even when hardware cannot be customised, software can be modified to support localisation needs.

The importance of localisation

While English has become the *lingua franca* of the international business world, there are over 6909 languages used in the world, as per *Lewis, M Paul (ed.), 2009. Ethnologue: Languages of the World, Sixteenth edition. Dallas, Tex. (http://www.ethnologue.com/)*. Huge populations are deprived of the full benefit of computing if the devices do not support their preferred languages. Without technology's help, there is a danger that these languages could become extinct. When a language dies, the rich cultural heritage of a segment of the world population disappears. Hence, localisation is important.

More and more operating systems and application software are being localised. This is opening up employment opportunities too, for people skilled in languages and computers.

The history of localisation

The origins of localisation can be traced to translation, where a speech or written document is translated from one language to another. Once computing devices became common, there was need to organise and customise the translation for software environments. That's how localisation was born. You can understand the growth of localisation with the example of the Debian distribution. Figure 2 shows the Debian versions and their language support, with a steady increase in the number of languages.

Debian Version	Code name	Release date	#of languages
3	woody	2002-07-19	40
4	etch	2007-04-08	58
5	lenny	2009-02-14	63
6	squeeze	2011-02-06	70

Figure 2: Debian releases and language support

While early localisers had to struggle with simple editors, specialised tools and platforms were later developed to support localisation.

Localisation vs internationalisation

Internationalisation is usually abbreviated to i18n (as there are 18 letters between the letters 'l' and 'n'). It denotes the packaging of the strings used in software so that the corresponding strings from a user's language can be deployed without impacting the functionality of the application. Apart from that, internationalisation is a set of practices followed by developers so that the application presents information and/or processes as per the expectations of the target users. Localisation, for most common usage, is concerned with translating the menu strings and application messages to the users' desired language. This requires a good knowledge of the target language, and a style guide for translation. For documentation localisation, a good command of the language and translation skills will also be needed.

As can be seen from the chart (Figure 3), localisation and internationalisation are closely interlinked, so work on both never ends. As long as software or the software environment changes, localisation work continues. This also explains why, while there are so many languages spoken in the world, localisation is limited to a smaller subset of them.

Localisation initiatives in Indian languages

IndLinux: The earliest localisation initiatives in India were from IndLinux, a Free Software group. IndLinux was started by Prakash Advani and Venkatesh Hariharan in December 1999 as a portal. Linux Bangalore/2001 was the first event when Indian Free Software groups got together to share developments on localisation. At Linux/Bangalore 2002, GNOME in Hindi was demonstrated. Various teams working with other Indian languages contributed to other language versions of GNOME, which resulted in the release

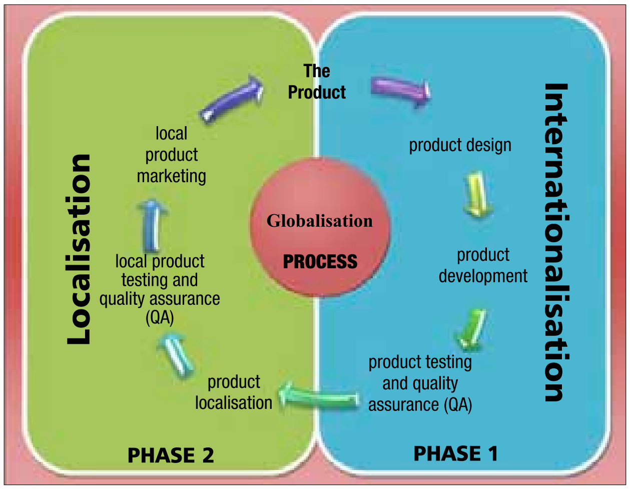


Figure 3: Internationalisation and localisation (http://commons.wikimedia.org/wiki/File:Globalisationchart)

of Rangoli 1.0 beta in 2005, supporting several languages. Other teams brought out language-specific Linux distributions around the same time. The language support became part of mainstream distributions like Fedora, Debian and Ubuntu over the years. When Ubuntu 11.04 was released on April 28, 2011, it included boot-time local support for the following Indian languages apart from English: Bengali, Gujarati, Hindi, Kannada, Malayalam, Marathi, Nepali, Tamil and Telugu.

FUEL: Frequently Used Entries for Localisation (FUEL) was initiated by Rajesh Ranjan to improve the quality of localisation, by standardising the translations for most common words or phrases. Twelve languages have been part of the initiative.

Microsoft's initiative: Windows 7 has been released with support for 95 languages. For Indian languages, Microsoft has a language portal called BhashaIndia, through which it enables users of its products to participate in localisation.

What's next?

Through this series of articles, I aim to provide information on various localisation platforms and tools, which can help readers to become active contributors in improving the language support for their preferred languages. I would appreciate your feedback at *arjunaraoc@ieee.org*.

For more information, visit:

http://indlinux.org/
https://fedorahosted.org/fuel/
http://bhashaindia.com/ END

By: Arjuna Rao Chavala

The author has over 25 years of experience working in the government, in private organisations and NGOs. Currently, he is an independent consultant in the areas of IT, program/engineering management and open source. Over the last four years, he has contributed to Firefox, Debian, and Ubuntu localisation, and font and keyboard enhancements for Indic languages. He can be reached through his website *http://arjunaraoc.blogspot.in/* and Twitter ID *@arjunaraoc*.

LOCALISATION

THE STATUS OF INDIC LOCALISATION

Having understood the history and growing importance of localisation in India, as well as the key local initiatives being taken in the previous article of this series, let us now look at the status of localisation in Indian languages, the current usage and the potential for growth.

Part—2

Let's first take a look at the status of localisation in India from the localiser's perspective. In order to quantify the status of localisation, we need to make a few choices about the software components. A computer user experiences several software, right from the moment the computer is switched on till it is switched off. These include boot and installation programs, the Graphical User Interface (GUI), a browser and several application software.

In the past, installation programs supported only English. After the introduction of GUIs and the progress of localisation, installation software (except for the initial boot screen) started sporting non-English languages. For the purposes of assessment, I've picked the Debian installer, as Debian and its derivatives have the maximum usage.

Many different GUIs are available in the free software world; GNOME and KDE are the most common, and GNOME is the most popular, which is why I've chosen to analyse the extent of its localisation. Of the many different browsers that are available, Firefox is the most popular, hence fits the bill for our analysis.

Software applications are specific to a task—whether preparing documents, analysing data, preparing presentations, working with pictures, playing music or video, etc. An office package that comes with a word processor, spreadsheet and presentation software has become perhaps the most common application. LibreOffice (a fork of OpenOffice) is the most common office software for free software users; hence, it has been selected as the fourth component for analysis.

For the 22 official Indian languages, the status of support for each of the selected software components, in their latest versions (in terms of release, beta, and availability) is given in Table 1.

When you carefully analyse the data, you will find that:

- Urdu has zero localisation.
- There are eight languages that have 100 per cent release-

Table 1: Localisation Status of the Main Components of Free Software

Language	India Speakers (Census 2001, in millions)	Wheezy Debian Installer Level 1	Gnome 3.4#	Firefox 11.0	Libreoffice 3.5.1#
Assamese	13	NA	Release	Release	NA
Bengali	83	Release	Release	Release	Release
Bodo	1.4	NA	NA	NA	Beta
Dogri	2.3	NA	NA	NA	Beta
Gujarati	46	Release	Release	Release	Release
Hindi*	258	Release	Release	Release	Release
Kannada	38	Release	Release	Release	Release
Kashmiri	5.5	NA	NA	NA	Beta
Konkani	2.5	NA	NA	NA	Beta
Maithili	12	NA	Beta	Beta	Beta
Malayalam	33	Release	Release	Release	Release
Manipuri	1.5	NA	NA	NA	Beta
Marathi	72	Release	Release	Release	Release
Nepali	2.9	Release	Beta	NA	Beta
Oriya	33	NA	Release	Release	Release
Punjabi	29	Release	Release	Release	Release
Sanskrit	0.01	NA	NA	NA	Beta
Santhali	6.5	NA	NA	NA	Beta
Sindhi	2.5	NA	NA	NA	Beta
Tamil	61	Release	Release	Beta	Release
Telugu	74	Release	Release	Release	Release
Urdu	52	NA	NA	NA	NA

* There are 422 million Hindi speakers, if you include those who speak its various dialects.
# A 75 per cent translation of the GNOME and LibreOffice UI is taken as usable localisation.

level coverage of all components.

- All Indian languages (barring Urdu) have release or beta versions of LibreOffice.
- Localisation efforts appear to have been prioritised in the following order: LibreOffice, GNOME, Firefox and Debian installer.
- Some Indian languages like Hindi, Tamil and Bangla are international in nature, and have benefited from worldwide contributors. The exception seems to be Urdu.

The user statistics

Localisation will improve only when there are users, so it is good to understand the user statistics. For FOSS, it is not easy to find out the extent of language use, as software is freely distributed. Firefox has a facility called Blocklist, which is used to ensure security for users by helping with warnings about suspicious sites, add-ons, etc. This feature records the preferences of Firefox users, each time they use the browser. Hence, the Firefox statistics are a good indicator of usage. From both the reading and editing perspective, the statistics of Wikipedia, the fifth-largest Internet site, have also been considered. Apart from this, bloggers constitute an important segment of language computing. The analysis of a bloggers' survey done by the *Indibloggers.in* portal has been used as another indicator. Table 2 shows the statistics of users for a sample of Indian languages.

A careful look at this table reveals that Indic users are a very tiny fraction of those who speak local languages. Hindi, Marathi, Tamil, Telugu, Malayalam and Bengali are ahead of the rest. Urdu registers its presence in Wikipedia and the bloggers' world.

Potential for growth

The recently released Census 2011 data on households with computers provides a rich source of information on the scope for using Indian languages. Of 246 million households, 9.5 per cent have computers, which amounts

Table 2: Users of localisation (partial statistics)

Language	Localised Firefox users in India as of 26 March 2012	Wikipedia worldwide Active Editors as of Feb 2012	IndiBloggers Statistics as of May 2009
Assamese		20	
Bengali		63	0
Bodo			
Dogri			
Gujarati		16	13
Hindi*	1928	62	316
Kannada	138	22	25
Kashmiri		0	
Konkani			
Maithili			
Malayalam	240	90	32
Manipuri		0	
Marathi	254	47	55
Nepali		21	
Oriya		9	
Punjabi		4	6
Sanskrit		18	
Santhali			
Sindhi		1	
Tamil	393	88	126
Telugu	284	38	47
Urdu		19	6

to 23.37 million people. The census also shows that 53.32 per cent of households have mobile phones, which means 131 million people. As smartphone prices reduce, we can expect households to migrate to tablets or smartphones with touchscreen interfaces. This will lead to a tremendous use of computing devices in India in the near future, as Indic language users will be able to comfortably get over the English language barrier that is associated with text-based input methods.

Now that you know more about where Indian language localisation stands, you can help by delving into the details of your preferred language(s), and contributing to promote and popularise Indian language computing. I look forward to your questions/feedback. END

For more information

L10n statistics:
- Debian installer: *http://d-i.debian.org/l10n-stats/*
- GNOME: *http://l10n.gnome.org/languages/*
- Firefox: *http://www.mozilla.org/en-US/firefox/all.html*
- LibreOffice: *https://translations.documentfoundation.org/*

Usage information:
- Firefox: *http://bit.ly/1YnP8L*
- Wikipedia: *http://stats.wikimedia.org/EN/Sitemap.htm*
- Indiblogger stats: *http://blog.indiblogger.in/2009/06/15/statistics-from-the-indian-blogosphere/*
- Household highlights: *http://www.censusindia.gov.in/2011census/hlo/hlo_highlights.html*

By: Arjuna Rao Chavala

The author is an independent consultant in the areas of IT, program/engineering management and open source. He is also the president of Wikimedia India and the WG Chair for the IEEE-SA project on 'Virtual keyboard standard for Indic languages'. He can be reached through his website *arjunaraoc.blogspot.in* and his Twitter ID *@arjunaraoc.*

Localisation The Gettext Framework

Having looked at Indian language localisation in the previous article of this series, let us now look at a major system for internationalisation called Gettext.

Part—3

Even if localisers work with Web-based environments, it is important that they have a good understanding of the underlying details to be able to appreciate the localisation message source string formats. Despite several initiatives to standardise internationalisation support across computer operating systems, it has not been achieved. If one needs to deliver software to run on multiple systems, a software-specific framework is used, and language packs are built specially for such software. The main examples of this kind of software are Firefox and LibreOffice. Let's explore locale briefly before getting into the details of Gettext system,the most common framework with support for various programming languages.

Locale

Locale is the term for the language and country specific practices for reporting date, time,currency etc of the user. A program written in English without any notion of localisation is designated to have the locale ‘C’, probably arising from the most popular system programming language 'C'. All software supporting localisation needs to operate as per the locale of the user. The locale basically consists of a two-letter language code and a two-letter country code, separated by an underscore (ll_CC). For example, it is *en_US* for US English and *te_IN* for the Indian language Telugu. When software with localisation support executes, it uses the locale definition in the environment to pick the relevant language message catalogs.

The Gettext system

Gettext was originally developed by Sun Microsystems in the early 1990s. GNU released its version in 1995. It is implemented for most programming languages and scripts. This includes a set of tools, guidelines, and a run-time library to utilise translations, plus a few independent programs to manipulate the various messages.

Any computer program has statements defining data, reading data and writing data, apart from computation on the same. An internationalisation system should be able to operate on the data without affecting the rest of the program, which means that the data definition and usage need to pass through special functions. At the same time, the use of special functions should not distract from the readability of the program. Gettext is the name of the function that is used for this purpose. A function whose name is a simple underscore (_()) is used as an alias for *gettext()*, to improve readability. A programmer has to internationalise a program, and then use the usual compilation path for building the program. Another path is required to extract translatable messages, localise and build the localised message set, as well as to place the same in the specified location. More extensive information on the guidelines for writing messages is available in the GNU Gettext manual. These deal with constant strings, plurals and information specifically relevant to the programmer, translator and maintainer.

A primer on internationalisation and localisation

Let’s take the simple ‘C’ language program ‘Hello World’, as in Listing 1, and look at how it can be internationalised and then localised. Note that the code has been tried out on Ubuntu 12.04, set up with English and Telugu locales, and a proper build environment for the GNU C compiler (GCC 4.6.3). Please ensure the same or a similar working environment before trying these programs.

Listing 1: hello.c

```
/*hello_English program*/
#include <stdio.h>
#include <stdlib.h>
int main() {
    printf("Hello World\n");
    return 0;
}
```

Internationalisation

Listing 2: hello18n.c

```
/*hello_internationalised program*/
#include <stdio.h>
#include <stdlib.h>
#include <libintl.h>
#include <locale.h>
```

```
#define _(String) gettext (String)
int main() {
   setlocale (LC_ALL, "");
   bindtextdomain ("helloi18n", "/usr/share/locale");
   textdomain ("helloi18n");
   printf (_("Hello World\n"));
}
```

The internationalisation instrumentation consists of adding header files (*libintl.h* and *locale.h*), and calling three functions at the start of the program. The *setlocale* function signals to the runtime that the locale as defined in the user environment needs to be used. The *bindtextdomain* function takes a package name and directory name as arguments. For a typical software project, there will be many source files that need to be compiled as a group, to build a package. Since we are compiling just one file, I have just used the name of the source file. The directory refers to the path where the message catalogues (the set of translated messages for the program) are available. The *textdomain* function is used to associate the filename for messages used in the program. The *printf* call is modified by placing the format string inside a function *_()*. The preprocessor will generate the gettext function calls. The localisation software *xgettext* (explained later) uses this as a keyword to extract translatable messages.

Now, let us compile and run the program:

```
$ gcc -o helloi18n helloi8n.c
Hello world
```

Localisation

Let us extract the strings, translate them, and prepare a binary so that the program can use another locale, using the Gettext library programs.

```
$ xgettext -k_ --package-name=helloi18n --package-version=1.0 -o
helloi18n.pot helloi18n.c
```

The parameter *-k_* specifies that we are interested in messages that are part of the function call with the underscore name. The package and version names are used in the header information. Usually, these are specified as part of the build files. As our example is fairly simple, I have used command-line arguments to pass these names. The option -o helps specify the file type as portable object template (pot) rather than the default portable object (po).

Listing 3: helloi18n.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: helloi18n 1.0\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2012-06-04 22:43+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"
"Language: \n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: 8bit\n"

#: helloi18n.c:15
#, c-format
msgid "Hello World\n"
msgstr ""
```

Listing 3 shows the *pot* file for our example program. It consists of a few header lines of information, followed by the source string and translated string, prefixed with *msgid* and *msgstr*, respectively. The first message is designed to give general information about the translation, such as the date, the name of the translator and team, etc. The subsequent lines provide messages used in the program. Each message follows the syntax as given in Listing 4.

Listing 4: Syntax for messages in a .po file.

```
white-space
#  translator-comments
#. extracted-comments
#: reference...
#, flag...
#| msgid previous-untranslated-string
msgid untranslated-string
msgstr translated-string
```

As the meaning of a word could be different based on the context, the above format provides options for the translator and programmer to provide additional context information. The file name and line number at which the message appears is provided as part of the reference. The flag is used to indicate the status or specific nature of translation. If it is fuzzy, it means that the translator needs to confirm the translation, as it was guessed from previous translation data. If it is obsolete, it could mean that it can be removed from the file. If it is C-format, it is an alert to the translator that this message is used in input or output statements, and follows the C language format and can have format information, newline information, etc. The previous-untranslated-string is updated by the *msgmerge* program (not described in this article), when creating a new pot file using previous translation files. Let's use *helloi18n.pot* and a version for the Telugu language with blank translations.

```
$ msginit -l te_IN -o te.po -i helloi18n.pot
```

This program will ask for the email of the user (given as *x@y.com* in this example), and use it as the contact address in the header. Now open the file in an editor (for example, *gedit*), and type in the translation for the string "*Hello World\n*" as "నమస్తేప్రపంచం\n" in the target language Telugu, and save the file. The resultant file is shown in Listing 5.

Listing 5: te.po corresponding to helloi18n.pot

```
# Telugu translations for helloi18n package.
# Copyright (C) 2012 THE helloi18n'S COPYRIGHT HOLDER
# This file is distributed under the same license as the helloi18n
package.
# arjun <x@y.com>, 2012.
#
msgid ""
msgstr ""
"Project-Id-Version: helloi18n 1.0\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2012-06-04 22:43+0530\n"
"PO-Revision-Date: 2012-06-04 22:54+0530\n"
"Last-Translator: arjun <x@y.com>\n"
"Language-Team: Telugu\n"
"Language: te\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"

#: helloi18n.c:15
#, c-format
msgid "Hello World\n"
msgstr "నమస్తేప్రపంచం\n"
```

Let's use the *po* file to build machine object (.mo) version, which is also called as message catalogue and store it in a specified location, with the filename changed to the package name. To do this, you may need superuser access as the standard location for storing such files has restricted access. If you do not have this, you can use your home directory, with appropriate replacements in the bindtextdomain function call of source code. The message catalogues are placed in a directory structure consisting of *language code/LC_MESSAGES* in the designated path. For example, Telugu messages are placed in */usr/share/locale/te/LC_MESSAGES/helloi18n.mo* and Hindi messages are placed in */usr/share/locale/hi/LC_MESSAGES/helloi18n.mo*.

This allows supporting additional locales by placing the localised message catalog files, without the need for recompiling the source, as the run-time library will automatically check for the translated message catalogues, and if it does not find any, will use the default messages. This also provides flexibility for the translation to progress independently.

```
$ msgfmt -o te.mo -v te.po
$ cp te.mo /usr/share/locale/te/LC_MESSAGES/helloi18n.mo
```

Results

Let's run the program with the default English locale, and then change the locale to Telugu. After which, let's run the program again, and then change the locale back to English. Note that the Telugu text rendering will not be perfect in a typical terminal, due to limitations in rendering the complex script of the language. You can direct the output to a file, and view the properly rendered message by opening that file in an editor. You can also use Terminator software (from software.jessies.org) which can display complex scripts properly. If you are using an older system, you can use *LANG* in place of *LANGUAGE* for the proper operation of the following code. Alternatively, you can change your language settings to the desired language, log out, and log in with the new set-up, and try the code.

```
$ echo $LANGUAGE
en_US.utf8
$ ./helloi18n
Hello World
$ export LANGUAGE=te_IN:te
$ ./helloi18n
నమస్తేప్రపంచం
$export LANGUAGE=en_IN:en
```

That concludes our primer on Gettext. We looked at how a typical 'C' language source can be instrumented to support internationalisation. We also looked at how the messages are extracted using *xgettext*, translated using *msginit*, and placed in the proper location, so that the run-time library can use them.

Several tool-kits, Web-based tools and standalone software were developed to allow the localiser to focus on translation without worrying about the nitty-gritty of files and formats. We will cover a few popular tools in forthcoming articles. END

Resources

- Introduction to Internationalisation Programming, Issue 103, November 2002, By Olexiy Ye Tykhomyrov in the Linux Journal, *http://www.linuxjournal.com/article/6176?page=0,0*
- Gettext manual: *http://www.gnu.org/software/gettext/manual/gettext.html*
- Terminator Cross platform GPL Terminal emulator with complex text rendering support. *http://software.jessies.org/terminator/*

By: Arjuna Rao Chavala

The author is an independent consultant in the areas of IT, program/engineering management and open source. He is also the president of Wikimedia India and the WG Chair for IEEE-SA project P1908.1 "Virtual keyboard standard for Indic languages". He can be reached through his website *http://arjunaraoc.blogspot.in* and Twitter id *@arjunaraoc*.

Correct positioning of the hands and fingers is very important to efficient typing. You will learn faster and type better if you follow the next recomendations.

Localisation
Localise an Application Menu

In this article, let us focus on localising an application by picking a typing tutor called Klavaro as an example. It is simple and requires to be localised for most Indian languages. Please note that the author has tried the tasks mentioned in this article on Ubuntu 12.04 with Telugu locale and you may need to adapt the instructions suitably for your locale/language of interest.

As Klavaro is not part of the default installation disk, it needs to be installed manually. Please select Ubuntu Software Centre, search for Klavaro and install it—or in a terminal, run *sudo apt-get install klavaro.*

For a localiser, it is important to explore the application to understand its functioning, and review how menus appear. The first screen you see when you run it is shown in Figure 1. You will notice that Klavaro has a simple menu structure at the bottom part of the screen. In the top portion, you will find a few big buttons, which lead to the basic steps of how to use the program. As these are examples of help files, containing text about how to use the application, I will cover such localisation for them in a future article.

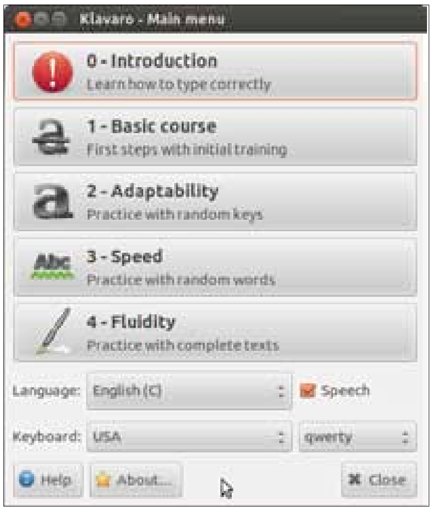
The first thing to do is to click 'About' to learn more about the application. From this, you will find the version number and a link to the site (*http://klavaro.sourceforge.net/*) where code and other information is available. On that site, you will find information about localisation (see Figure 2) and also the platforms or environments in which the localisation is coordinated. From the Web page, we learn that it uses Gettext for i18n and Translation Project (*http://translationproject.org/*) as the platform for l10n.

One also finds Poedit as part of the sample tools suggested for localisation, so let's use it to localise this package. Take a moment to check the status of localisation for your language of interest at the Translation Project. You may find that it is either not translated, is partially done or is fully translated. If it is not translated, you can download the *pot* file from the package host site or from the Translation Project site, and start localisation. In case you find the translation completed for your language, you can review the localised application and identify any improvements that are needed. Then you can download the current localisation file (*.po*) from the Translation Project, and continue with the following process, to improve it.

Poedit

Poedit needs to be installed manually; a simple *sudo apt-get install poedit* will do it. Run Poedit and you will be prompted to enter a few details that will be useful as you work with the tool—your user name, email ID, etc. If you are already using Ubuntu with your locale, you can get the benefit of generating the translation memory from your current set-up—select the *'Translation Memory'* tab in *Preferences* and add your language; then select *Generate Database*. A screen (Figure 4) with suggested paths of localisation catalogues will be shown. If you continue, all *.mo* files in those directories will be read, and a translation memory database will be created.

Poedit has three panes: the top one shows the messages catalogue (both the source language message in English, and the target language message, if already translated),


Klavaro - Main menu

0 - Introduction
Learn how to type correctly

1 - Basic course
First steps with initial training

2 - Adaptability
Practice with random keys

3 - Speed
Practice with random words

4 - Fluidity
Practice with complete texts

Language: English (C) ✓ Speech

Keyboard: USA qwerty

Figure 1: Klavaro with the English UI

the second pane shows the source string, and the third one shows the target language message edit box. The status row at the bottom presents the statistics of the file like *% translated*, *total strings*, *fuzzy strings*, etc.

Rename the downloaded pot file or *po* file as *klavaro<<languagecode>>.po* (for Telugu, *Klavarote.po*) and open it in Poedit. You will find the panes populated with information from the file. If your translation is zero per cent done, it is better to seed the localisation with messages that are in use from the previous localisations of other applications, if there is a match. Select '*Automatically translate using translation memory*' from the '*Catalogue*' menu. You will find some strings automatically translated, but marked as *fuzzy*. Scroll down to see the fuzzy translations. A moment's study will help you identify the correct and incorrect translations (see Figure 5).

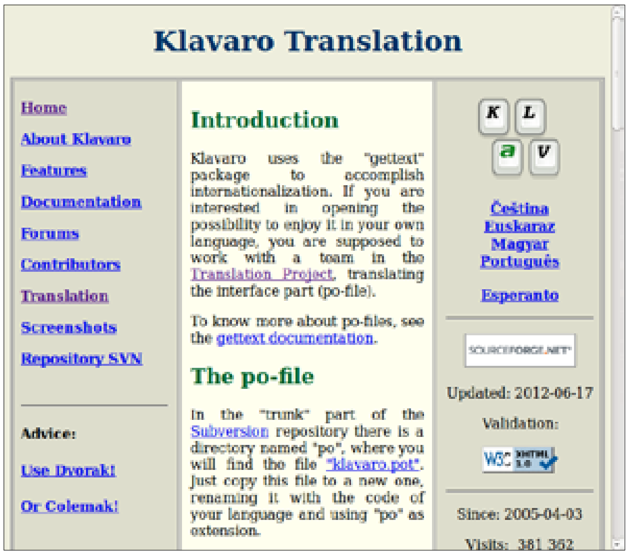
Update the project settings with the package name and other details (see Figure 6). You may need to browse the plural forms site (*http://translate.sourceforge.net/wiki/l10n/pluralforms*).

Now let us just focus on the UI elements in the launch screen. You can search for these strings, review the existing fuzzy translations, if any, and update the translation (see Figure 7). After doing this, save the file and exit the application. A *.mo* file is automatically generated when you close the file being edited. Now copy the *.mo* file to the appropriate location, as *klavaro.mo*, using the following command (you will need administrator privileges; this uses the location for Telugu):

```
$sudo mv klavarote.mo /usr/share/locale/te/LC_MESSAGES/klavaro.mo
```

After this, relaunch the application and Klavaro will be in your locale (see Figure 8 and compare with Figure 1). Please verify the localisation, and repeat the process if any mistakes are found which need to be fixed. In this manner, localisation can be improved to cover all messages.

After confirming that the localised user interface is working locally, you need to perform the important step of placing the localised files in the upstream for this project (this means the Translation Project). For this, you need to contact the localisation coordinator listed for your language at the Translation Project and become a member of the language team. Note that it is important to disclaim any rights over your translations, so that


Klavaro Translation

Introduction

Klavaro uses the "gettext" package to accomplish internationalization. If you are interested in opening the possibility to enjoy it in your own language, you are supposed to work with a team in the Translation Project, translating the interface part (po-file).

To know more about po-files, see the gettext documentation.

The po-file

In the "trunk" part of the Subversion repository there is a directory named "po", where you will find the file "klavaro.pot". Just copy this file to a new one, renaming it with the code of your language and using "po" as extension.

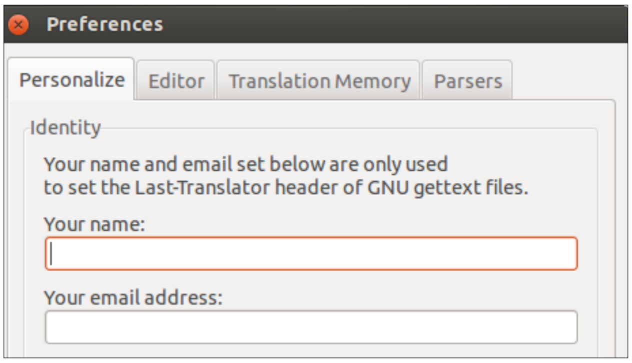
Updated: 2012-06-17

Validation:

Since: 2005-04-03

Visits: 381 362

Figure 2: The Klavaro website, translation page


Preferences

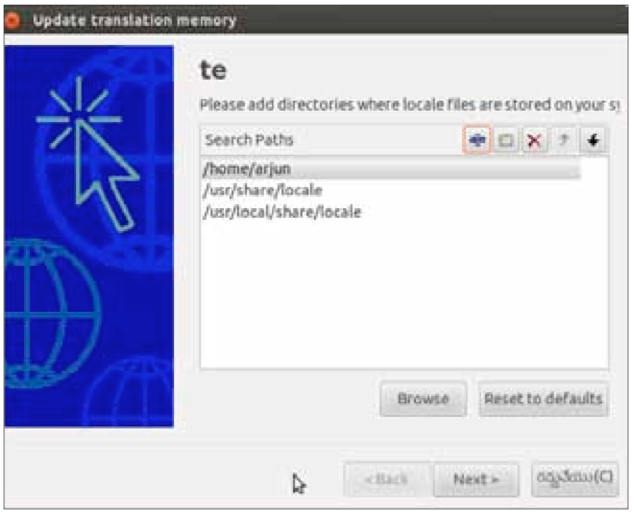
Identity

Your name and email set below are only used to set the Last-Translator header of GNU gettext files.

Your name:

Your email address:

Figure 3: Poedit preferences


Update translation memory

te

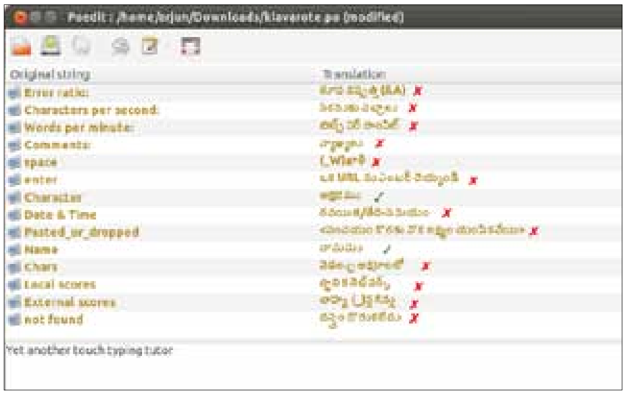
Please add directories where locale files are stored on your s

Search Paths

/home/arjun
/usr/share/locale
/usr/local/share/locale

Figure 4: Poedit translation memory set-up

they may be reused without any constraints, by sending a signed declaration to the Translation Project coordinator. Traditionally, this entailed sending a scan of a paper certificate by email and waiting for the response, which, at times, stretched to a few

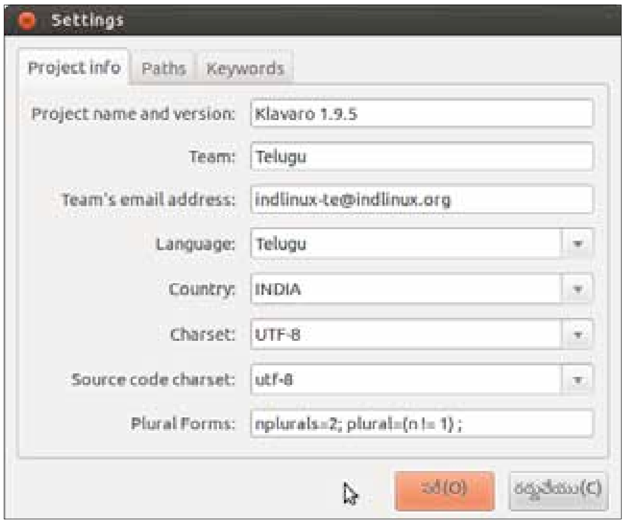


Poedit : /home/arjun/Downloads/klavarote.po (modified)

Original string	Translation
Error ratio:	[illegible] (ĖA) ✗
Characters per second:	[illegible] ✗
Words per minute:	[illegible] ✗
Comments:	[illegible] ✗
space	(_W[illegible] ✗
enter	[illegible] URL [illegible] ✗
Character	[illegible] ✓
Date & Time	[illegible] ✗
Pasted_or_dropped	[illegible] ✗
Name	[illegible] ✓
Chars	[illegible] ✗
Local scores	[illegible] ✗
External scores	[illegible] ✗
not found	[illegible] ✗

Yet another touch typing tutor

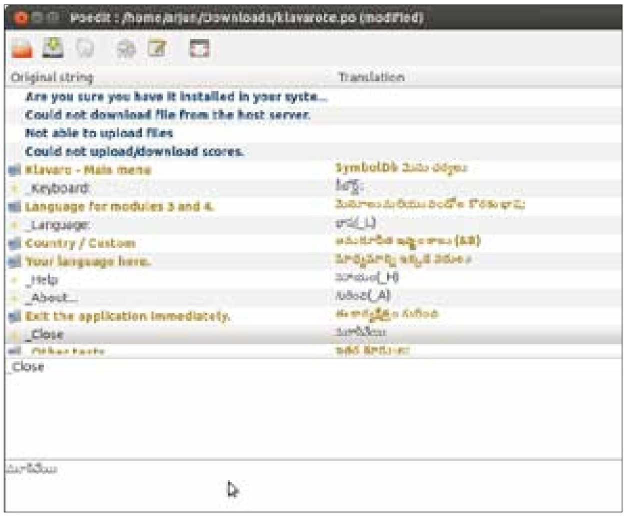
Figure 5: After automatic translation (tick-mark annotations)



Settings

Project name and version:	Klavaro 1.9.5
Team:	Telugu
Team's email address:	indlinux-te@indlinux.org
Language:	Telugu
Country:	INDIA
Charset:	UTF-8
Source code charset:	utf-8
Plural Forms:	nplurals=2; plural=(n != 1);

Figure 6: Project settings



Poedit : /home/arjun/Downloads/klavarote.po (modified)

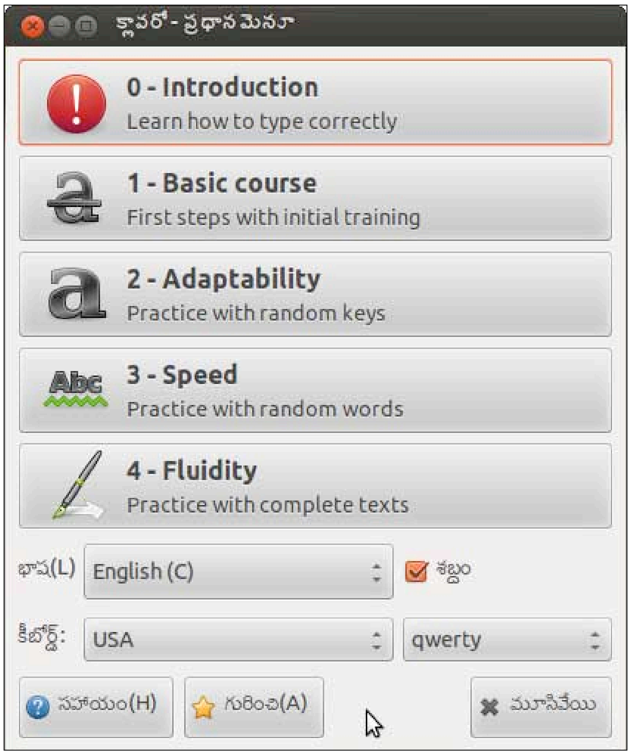
Original string	Translation
Are you sure you have it installed in your syste...	
Could not download file from the host server.	
Not able to upload files	
Could not upload/download scores.	
Klavaro - Main menu	SymbolDb [illegible]
_Keyboard:	కీబోర్డ్:
Language for modules 3 and 4.	[illegible]
_Language:	[illegible](_L)
Country / Custom	[illegible] (ĀB)
Your language here.	[illegible]
_Help	[illegible](_H)
_About...	[illegible](_A)
Exit the application immediately.	[illegible]
_Close	[illegible]
Other texts	[illegible]

_Close

[illegible]

Figure 7: Completed translations for the launch screen menu

months. This process is now being Web-enabled to make it easy for users. Then you need to send an email with the *package-version.language.po* on the subject line and the localised file as an attachment to *robot@translationproject.org*. If the robot



క్లావరో - ప్రధాన మెనూ

0 - Introduction
Learn how to type correctly
1 - Basic course
First steps with initial training
2 - Adaptability
Practice with random keys
3 - Speed
Practice with random words
4 - Fluidity
Practice with complete texts

భాష(L): English (C) — ✓ శబ్దం

కీబోర్డ్: USA — qwerty

సహాయం(H)
గురించి(A)
మూసివేయు

Figure 8: Klavaro with the Telugu UI

responds with any errors, those need to be addressed before attempting to send the attachment again. Once the attachment is accepted, your localisation will be picked up by all Linux distributions and will reach the end users in due course of time.

Note: As multiple input methods are used in the Indian language alphabet keyboard, short cut keys of English should be retained in the localisation by adding them in parentheses. For example, "*_Language:*" becomes localised as *<localised name for Language>(_L)*

There are several other standalone tools like Gtranslator, Virtaal and Lokalize, which can be used for localisation. Lokalize is a bit more advanced, as it shows the progress of localisation graphically for the entire set of PO files for an application and provides additional features like glossary support. We will explore it in forthcoming articles in this series. Please try localising any software of interest and write in with your questions on any aspect of localisation. END

By: Arjuna Rao Chavala

The author is an independent consultant in the areas of IT, program/engineering management and open source. He is also the president of Wikimedia India and the WG Chair for the IEEE-SA project P1908.1 "Virtual keyboard standard for Indic languages". He can be reached through his website *http://arjunaraoc.blogspot.in* and his Twitter ID: *@arjunaraoc*.

Localisation

Localise Application Help Files

This article focuses on the nuances of localising application help files.

Online help and User manuals are given lesser priority when localising software. This is usually due to complexity of the help/documentation and also inadequate understanding of dealing with the relevant files. Most help documentation consists of multiple pages written in some form of markup language with each page consisting of paragraphs and links to other pages. XML based mark-up languages have traditionally been used for help documentation. There are helpful utilities to generate documentation in HTML, PDF and other online formats.

In this article, I present the structure of the help files, explore Mallard project for online help and illustrate the sequence of steps for localising with Gedit as an example.

The structure of help files

On a typical UNIX system, help files are stored as follows:

```
/usr/share/help
  |--/gedit : Application
     |----/C :Reference help in English-US
     |----/te : Langauge help inTelugu
     |----/hi : Language help in Hindi
```

Under each help directory, there is a set of mark-up files and a directory for figures. Each mark-up file can be converted into a '.po' file, which can be translated in a localisation environment and converted back to XML. XML utilities are used for these steps. As there can be several XML files, there are also tools available to convert all of them into a single '.po' file, which can be converted back to XML files after translation.

After translation, the XML files and figures are copied into the designated location, so that the system's help engine can pick them up as per the locale of the user. For error checking, the translated files can be viewed in the help browser, or can be processed to yield HTML or PDF files.

Mallard

Mallard is gaining momentum as a mark-up language better suited for the purposes of online help. Its chief advantage is reference inversion, i.e., each page specifies the names of other pages that are to be linked to it. The Mallard compiler ensures that help pages are updated with proper links from the source pages. This makes the documentation evolve with the software, with changes required to fewer files. A lot of disadvantages of the older Docbook format have also been overcome with Mallard.

Process for gedit localisation

Let's look at how to localise the Gedit help into Telugu. Gedit is a popular and powerful editor in GNOME environment. It supports internationalized text (UTF-8) and features configurable syntax highlighting for various languages along with print preview, configurable fonts. Figure 1 shows the Telugu version of Gedit (localised) and Figure 2 shows its help page (in English). Install *itstool* (on Ubuntu, *sudo apt-get install itstool*). If any of the commands in the following steps don't work, install the corresponding packages and repeat the steps.

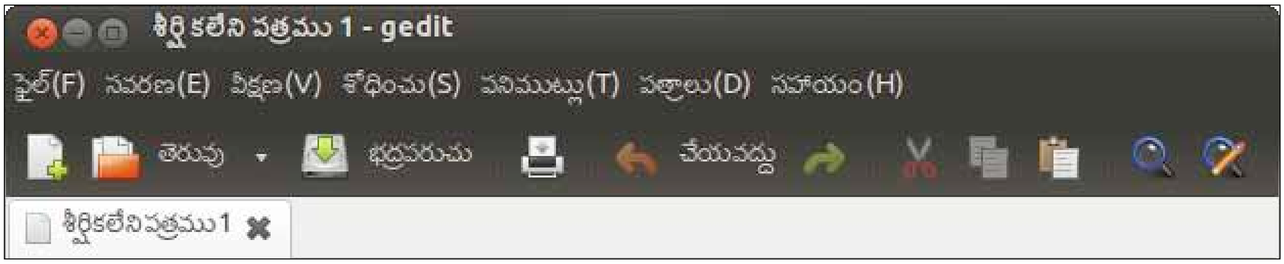
Step1: Locate the reference Gedit documentation on your system as per the details given earlier, or clone the repository with *git clone http://git.gnome.org/browse/gedit*.

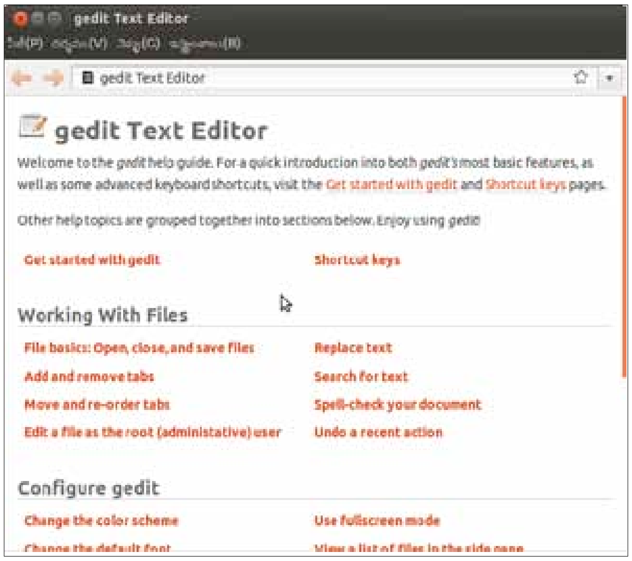
Step 2: Build *te.po* from the reference help files. The following commands are for the local version of the Gedit documentation repository, if the Telugu PO file is not already part of the package: "at /usr/share/help/te".

Note: I picked the Gedit application as an example for this, as it is the default text editor of GNOME. I have tried out everything mentioned in this article on Ubuntu 12.04 and configured it for Telugu. You may need to adapt the steps for your OS and locale.

```
$ cd gedit/help/C
$ itstool -l te *.page -o ../te/te.po
```

Step 3: Translate the PO file. Use Poedit or any PO file editor or the Gedit text editor itself to translate. Make use of the translation memories and glossaries available for your language to enhance productivity and consistency. Keep Gedit open in the desired locale, and take care to use the localised UI in the help documentation. One easy way to ensure this is to generate translation memory and glossary from the localised UI po file, and to use localisation tool such as poedit to provide translation tips. As the strings may


శీర్షికలేని పత్రము 1 - gedit
ఫైల్(F) సవరణ(E) వీక్షణ(V) శోధించు(S) పనిముట్లు(T) పత్రాలు(D) సహాయం(H)
తెరువు
భద్రపరుచు
చేయవద్దు
శీర్షికలేనిపత్రము1
Figure 1: Gedit with Telugu UI


gedit Text Editor

gedit Text Editor

Welcome to the gedit help guide. For a quick introduction into both gedit's most basic features, as well as some advanced keyboard shortcuts, visit the Get started with gedit and Shortcut keys pages.

Other help topics are grouped together into sections below. Enjoy using gedit!

Get started with gedit | Shortcut keys

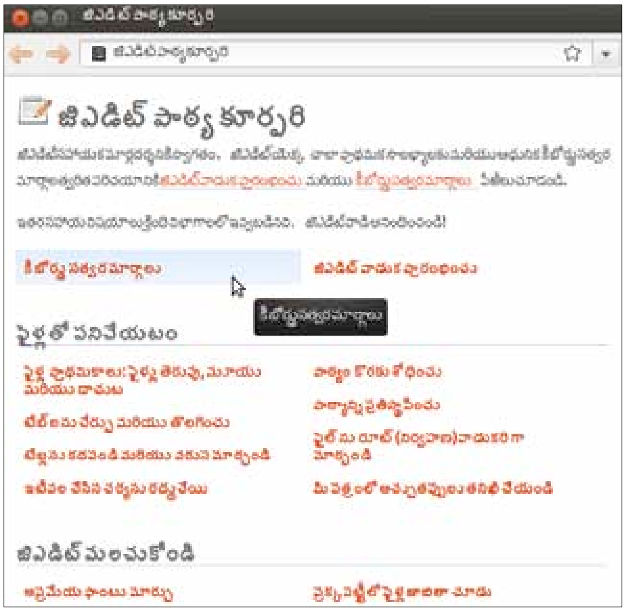
Working With Files

File basics: Open, close, and save files | Replace text
Add and remove tabs | Search for text
Move and re-order tabs | Spell-check your document
Edit a file as the root (administative) user | Undo a recent action

Configure gedit

Change the color scheme | Use fullscreen mode
Change the default font | View a list of files in the side pane

Figure 2: Gedit Help in English


జిఎడిట్ పాఠ్య కూర్పరి

జిఎడిట్ పాఠ్య కూర్పరి

ఫైళ్లతో పనిచేయుటం

జిఎడిట్ మలచుకోండి

Figure 3: Gedit Help in Telugu

be large in number, it is a good practice to search for the strings in the displayed root page of the help documentation and translate them first, before going on to other pages as per the links from the root page.

Step 4: Generate the MO file from the PO file (*msgfmt -v te.po -o te.mo*)

Step 5: Generate XML files with the translated MO file as input using (*itstool -l te -m te.mo -o testte/ *.page*). If *itstool* throws up errors, please check the PO file for syntactical correctness, and use the *msgmerge* utility to merge the translated PO file with the original PO file to improve the syntax of the localised PO file

Step 6: Copy the generated mark-up files to the location used by the help system for the application (*cp -r testte /usr/share/help/te/gedit)*. Update the figures with those corresponding to the target locale.

Step 7: View the help in the GNOME Help browser (*yelp /usr/share/help/te/gedit*), or launch the application and verify the quality of the translation.

Step 7: If required, repeat Steps 3-7.

Step 8: Submit the updated PO file as a comment to the Web-based GNOME documentation page for Gedit (*eg: http://l10n.gnome.org/vertimus/gedit/gnome-3-4/help/te*)

Figure 3 shows the help localised into Telugu.

We have explained the structure of the help files and illustrated the steps for localisation using Gedit as an example, which uses Mallard for online help. Help/User manual localisation needs to be completed soon after UI localisation to increase adoption of native language computing environments by users. END

For more information

[1] How to write a manual for a GNOME application with DocBook—Manuel Rego Casasnovas, *http://people.igalia.com/mrego/mswl/ils/how_to_write_a_manual_for_a_gnome_application_with_docbook.html*
[2] The Mallard website, *http://projectmallard.org/*
[3] The Itstool website, *http://itstool.org/*
[4] Gedit documentation links on the Gnome Mallard page *https://live.gnome.org/DocumentationProject/Tasks/ApplicationHelp*

By: Arjuna Rao Chavala

The author is an independent consultant in the areas of IT, program/engineering management and open source. He is also the president of Wikimedia India and the WG Chair for the IEEE-SA Project P1908.1, 'Virtual keyboard standard for Indic languages'. He can be reached through his *website http://arjunaraoc.blogspot.in* and *Twitter id @arjunaraoc.*

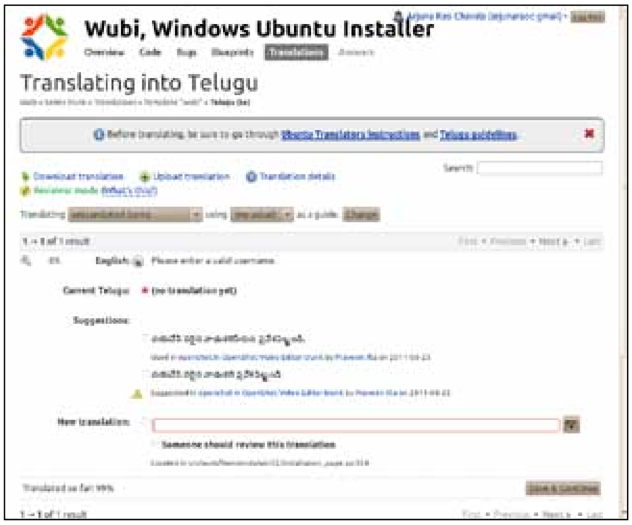
Web-based Platforms for Localisation

Previous articles in this series have explored how to localise the application user interface using Desktop software. This article covers localising using Web-based platforms, which are easier and convenient, as contributors do not have to worry about file formats, translation memories, version management systems, etc. These are expressly designed as Translation Management Systems, allowing multiple team members to collaborate on a project.

The features of web based platforms include display of localisation status and statistics of all the files of a project, support for translation memories and workflow management. Glossaries are also supported in some of the tools. Let us look at Launchpad.net, Pootle for LibreOffice, and Translatewiki.net as three examples.

Launchpad.net

This is one of the well-known Web-based localisation platforms. On visiting the site, you will learn that it supports 334 languages and has 64,938 translators in 43 translation groups. It was started by Canonical in 2005, and released as open source in 2009. Its translation tool is called Rosetta. It is much more than a localisation platform, as it supports code hosting with version control, release management, bug tracking, mailing lists and wikis. The translation feature provides a summary view of the status of localisation, and features suggestions from the vast translation memory. It can be used for localising operating systems and independent projects. If a package uses Launchpad.net as the upstream (the root place for managing the project), users can just do the localisation. If the upstream is different, then, after completing the localisation,



Wubi, Windows Ubuntu Installer

Translating into Telugu

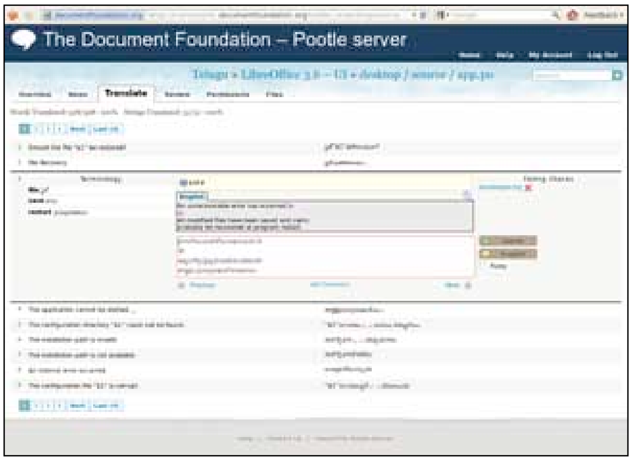
Current Telugu: (no translation yet)

Suggestions:

New translation:

Someone should review this translation

Figure 1: Using Launchpad.net to localise WUBI



The Document Foundation – Pootle server

Figure 2: LibreOffice localisation using Pootle

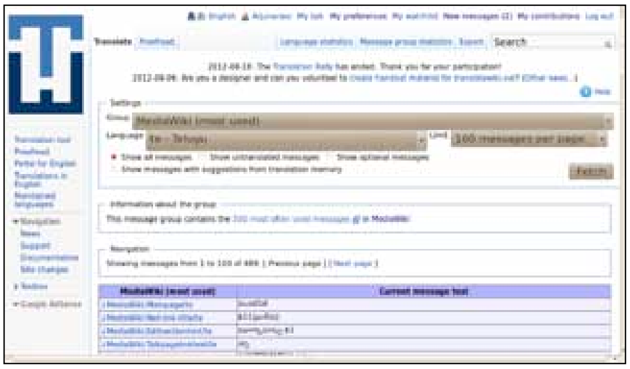


Figure 3: Localisation of MediaWiki using Translatewiki.net

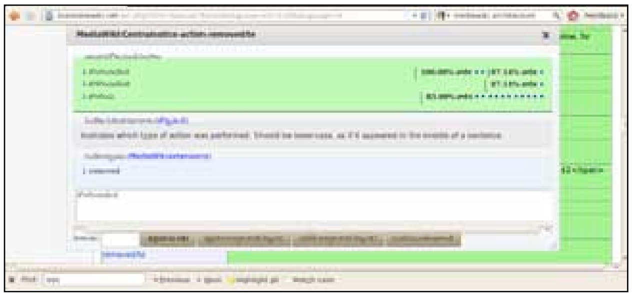


Figure 4: Message localisation overlay in Translatewiki.net

users have to download what they have localised and submit it for updating the upstream, separately.

Users have to create a launchpad.net account to contribute to localisation. It is advisable for users to become members of the localisation teams to collaborate actively with other members. All the localisation suggestions submitted by users are reviewed by the translation coordinator. The localisation pages offer different options to filter the messages for improved productivity. Figure 1 shows the Launchpad screenshot for the localisation of the Windows Ubuntu Installer (WUBI) into Telugu. You can see the different menus available on a translation page. The messages are presented as one set, even though they may be present in different source files.

Pootle

LibreOffice uses Pootle (*PO-based Online Translation / Localisation Engine* tool) for Web-based translation. Pootle is developed by *Translate.org.za*, a South Africa-based non-profit organisation focused on the localisation of open source software. The Web-based tool leverages the Translate Toolkit software from the same organisation, and complements a standalone localisation tool called Virtaal. Many open source projects are hosted on Locamotion website (*http://pootle.locamotion.org*). To localise a project, an account is needed on the server. After logging in, languages and projects of interest need to be configured. If required, the leaders of language projects need to be contacted to get the necessary permissions.

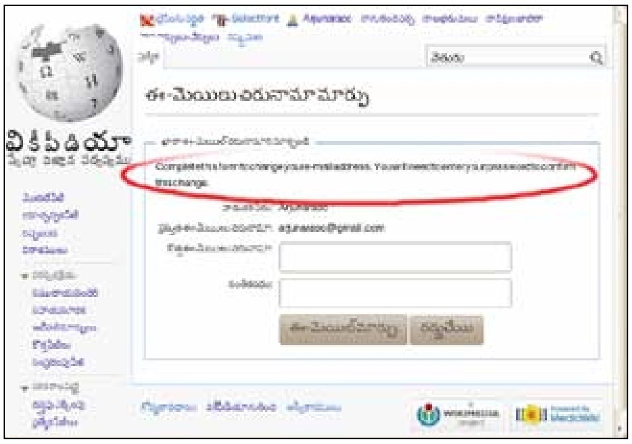


Figure 5: A page in Telugu Wikipedia with some strings that are not localised

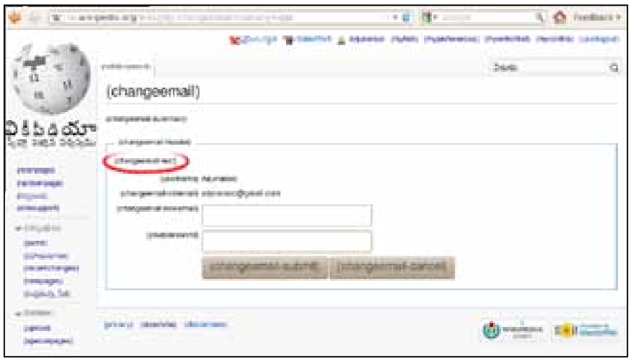


Figure 6: Displaying message_ids corrresponding to the screen in Figure 5

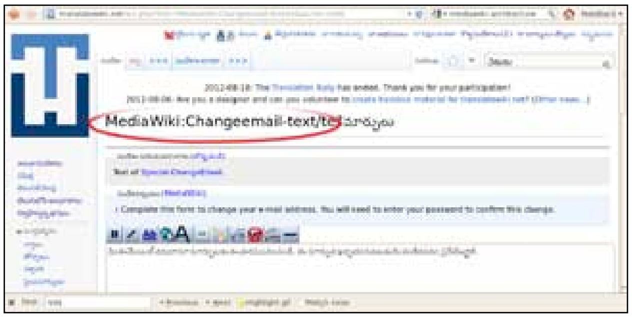


Figure 7: Translating the specific message of MediaWiki in Translatewiki.net

Figure 2 shows a sample screenshot of the Telugu localisation for LibreOffice, hosted on an independent instance of Pootle by the Document Foundation. The glossary suggestions are given on the left of the terminology files of the project on the server. The existing translation is checked for localisation defects, and the status is shown on the right side. This feature is unique to Pootle. Localisation text can be updated making use of the glossary suggestions. There is a provision to submit the update for review, as well as to mark a localised message as 'fuzzy' and add appropriate comments for review by others. Messages are organised as per the directory and file structure of the source.

Continued on Page no 55

part of a process. Typically, I write a shell script. Part of the process might involve using LibreOffice as mentioned above to convert the Excel spreadsheet to a CSV. Perhaps I extract some of the fields from the CSV, use them to interrogate a database, match the output against some other criteria, and finally .

I can send the output CSV to my customer because Excel accepts CSV as input. I suppose if it were requested, I could use LibreOffice to convert the *.csv* to *.xls* or *.xlsx*, but I have never been asked for that.

I find that scripting is much more powerful than many the things one might try to do in Excel. The process can be made more automatic, which is particularly advantageous if the process has to be run frequently or regularly. In the latter case, I can invoke my script from *cron*.

But do bear in mind that I am talking about tools. It's always best to find the most appropriate tool for the task. I haven't covered everything, but with this arsenal I can go a long way.

My previous article (Funny Mythness) discussed ways in which you could have Linux in the workplace using virtual machines. In this article, I've suggested ways in which you can move more towards Linux. I've talked about applications that allow you to inter-operate with others who are probably restricted to Microsoft's office applications; and I've discussed tools that allow you to convert from inconvenient formats to those more amenable to manipulation using Linux tools.

In the next article, I will look at UNIX tools and how to unleash their power on files converted with the tools in this article. END

By: Henry Grebler

The author has spent his days working with computers, mostly for computer manufacturers or software developers. His early computer experiences include relics such as punch cards, paper tape and mag tape. His darkest secret is that he has been paid to do the sorts of things he would have paid money to be allowed to do. Just don't tell any of his employers.

He has used Linux as his personal home desktop since the family got its first PC in 1996. Back then, when the family shared the one PC, it was a dual-boot Windows/Slackware set-up. Now that each member has his/her own computer, Henry somehow survives in a purely Linux world.

He lives in a suburb of Melbourne, Australia.

(Continued from Page 49....)

Translatewiki.net

This is a clever project that is in the process of localising MediaWiki. The Translate MediaWiki extension was started in 2006 by *translatewiki.net/wiki/User:Nike* and *translatewiki.net/wiki/User:Gangleri*, and has become very popular for localising MediaWiki, its extensions and a host of other software. MediaWiki itself has been designed with internationalisation and localisation in mind. So it's no wonder that it is available in over 280 languages. Like other Web-based projects, this too requires a contributor to create an account and select preferred languages. Then a project can be selected for localisation, and its various messages can be localised.

Figures 3 and 4 show a typical localisation screenshot. The suggestions from translation memory are shown, along with the percentage of matches, and the blue dots following the number are hyperlinks to the actual translation. The information about the context of the string is shown next, followed by the source, and then an edit box to enter the localised text. This update is saved just like a Wikipedia page. Additional menu buttons allow easy navigation across the messages. Another advantage with this tool is that most updates will go live on the respective language wiki projects of Wikimedia Foundation in 24 hours.

Most Indian-language MediaWiki projects would have been already localised. But occasionally, users may come across pages for which some messages are still in English (Figure 5). Usually, when a project is partially localised, try for typical free software. In MediaWiki, it becomes very simple to locate the message_id by *use* to the URL of the page. In such a case, MediaWiki software displays message_ids rather than actual English strings (Figure 6). Then you can use the message_id to search Translatewiki.net to directly locate the untranslated message string and translate it (Figure 7).

In this article, we have seen three different Web-based platforms for localisation, which make it easy for anybody to contribute to localisation in their preferred languages. So why not try your hand at it, and share your experience with fellow readers? I will be happy to devote an article exclusively to your feedback and questions. END

For more information

[1] Launchpad home page: *https://translations.launchpad.net/*
[2] Pootle features: *translate.sourceforge.net/wiki/pootle/features*
[3] LibreOffice translations using Pootle instance: *https://translations.documentfoundation.org/*
[4] Translatewiki.net home page: *translatewiki.net/wiki/Main_Page*

By: Arjuna Rao Chavala

The author is an independent consultant in the areas of IT, program/engineering management and open source. He is also the president of Wikimedia India and the WG Chair for the IEEE-SA project P1908.1, 'Virtual keyboard standard for Indic languages'. He can be reached through his website *arjunaraoc.blogspot.in* and Twitter ID *@arjunaraoc*.

Localisation

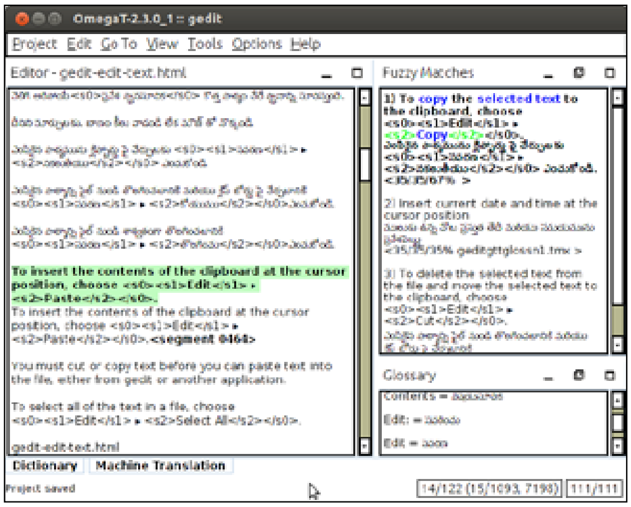
Localising User Documentation

Previous articles in this series have covered how to localise applications, using desktop and Web-based software. Most software have user documentation, which needs to be localised. This article describes how to use OmegaT, a free computer-aided Translation Memory (TM) tool, to translate documentation.

User manuals are more difficult to localise than help files or the user interface, as they have a lot more text and visuals to help people learn. While it is possible to copy the English version and create a local-language version in a simple editor, making modifications and generating newer versions is not easy. A specialised computer-based translation aid like OmegaT is a good tool for this purpose. Other proprietary but free-to-use tools, like the Google Translator toolkit, can also be used with certain limitations.

OmegaT is a free TM application written in Java, suitable for professional translators. It supports multiple-file projects, multiple translation memories, glossaries and fuzzy matching. It supports more than 30 document formats, including plain text, *.po* files, Microsoft, OpenOffice and various mark-up languages. It has a built-in spell checker, and an interface to Google Translate. It supports exporting to the various formats used in localisation tools.

Figure 1 shows a sample screenshot of OmegaT. It consists of three sub-windows; the first, on the left, is called Editor—it shows the source components, and also allows updating the corresponding target language translation. The second one, at top right (the match viewer), shows matches from translation memory. The third one, at bottom right, is a glossary viewer that shows the glossary matches.



```
[illegible]
To insert the contents of the clipboard at the cursor
position, choose <s0><s1>Edit</s1> ▸
<s2>Paste</s2></s0>.
To insert the contents of the clipboard at the cursor
position, choose <s0><s1>Edit</s1> ▸
<s2>Paste</s2></s0>.<segment 0464>

You must cut or copy text before you can paste text into
the file, either from gedit or another application.

To select all of the text in a file, choose
<s0><s1>Edit</s1> ▸ <s2>Select All</s2></s0>.

gedit-edit-text.html
```

```
1) To copy the selected text to
the clipboard, choose
<s0><s1>Edit</s1> ▸
<s2>Copy</s2></s0>.
[illegible]
<35/35/67% >

2) Insert current date and time at the
cursor position
[illegible]
<35/35/35% geditgtglossn1.tmx >

3) To delete the selected text from
the file and move the selected text to
the clipboard, choose
<s0><s1>Edit</s1> ▸
<s2>Cut</s2></s0>.
[illegible]
```

```
Contents = [illegible]
Edit: = [illegible]
Edit = [illegible]
```

```
Project saved                    14/122 (15/1093, 7198)  111/111
```

Figure 1: OmegaT for translating documents

OmegaT operation is based on projects. For each project, the user needs to specify the source language, target language, basis for segmentation (either sentence or paragraph), and the directories for source files, target files, translation memory and glossary files. Once these are specified, all the source files are imported into the project. The user can select a file and do the translation. The translation memory is updated with each change made to the translation. To verify the translation, select *Create Translated Documents* from the *File* menu. The target directory will be updated with the translated files. Those can be processed further if required and viewed with an appropriate tool.

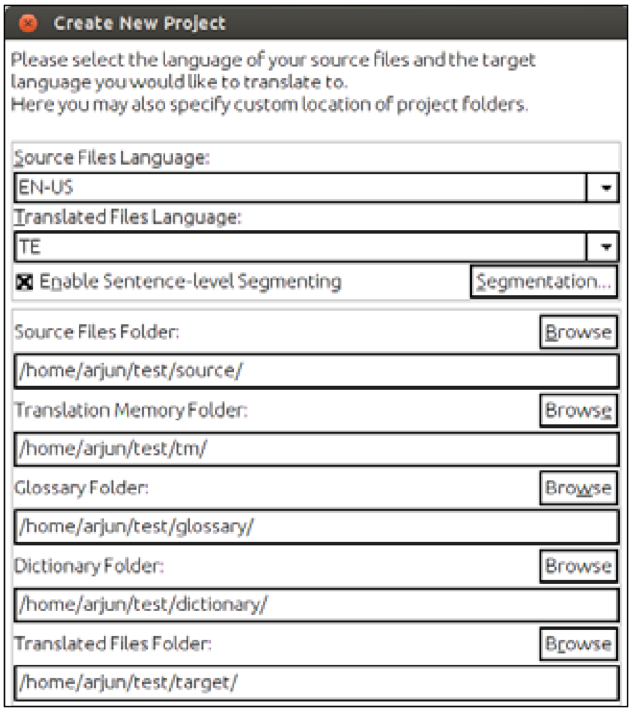
In order to get a high-quality translation, it is important that the translation memories generated from the user interface and help file localisation are used in OmegaT.

Gedit manual translation

The use of this tool can be demonstrated by localising the Gedit manual from English to Telugu.

The Gedit manual was originally written in HTML. Recently, it is being rewritten using Mallard, as I explained in a previous article. For the purpose of this article, let's use the HTML version. The manual is available for download from the GNOME documentation website *(http://library.gnome.org/users/gedit/)*. Download version 2.30.4 to get the HTML version. To populate the translation memory, the Telugu *.po* file can be downloaded from the GNOME localisation site *(http://l10n.gnome.org/vertimus/gedit/master/po/te)*. With the use of localisation tools like Virtaal, the translation can be exported to the TMX format for use in OmegaT. A glossary can be updated by extracting all the one- and two-word strings from the user interface, using Translate Toolkit's utilities and stored in a CSV format, suitable for OmegaT.

Create a new project (Figure 2), with segmentation set to sentence boundaries; place the relevant source, translation memory and glossary files in the directories specified in the project—and the set-up is complete. Now, you can select specific files from 'Project files' (Figure 3) for translation. Each source segment, and its place-holder for the translation, is shown in the



Create New Project

Please select the language of your source files and the target language you would like to translate to.
Here you may also specify custom location of project folders.

Source Files Language: EN-US

Translated Files Language: TE

☒ Enable Sentence-level Segmenting [Segmentation...]

Source Files Folder: [Browse] /home/arjun/test/source/

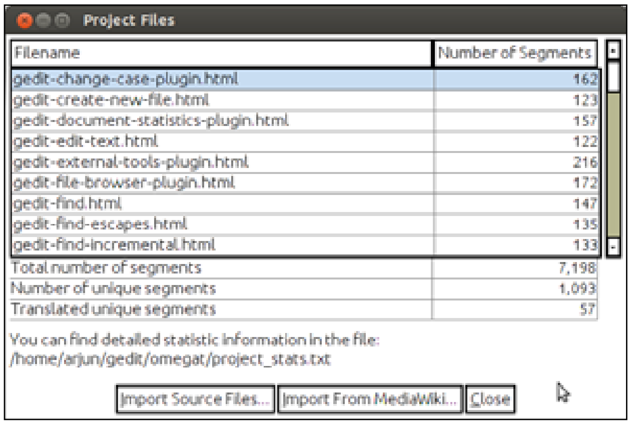
Translation Memory Folder: [Browse] /home/arjun/test/tm/

Glossary Folder: [Browse] /home/arjun/test/glossary/

Dictionary Folder: [Browse] /home/arjun/test/dictionary/

Translated Files Folder: [Browse] /home/arjun/test/target/

Figure 2: OmegaT new project properties

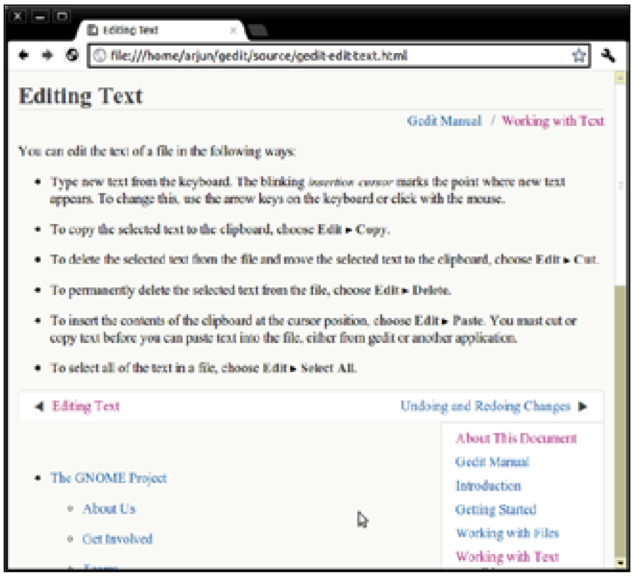


Project Files

Filename	Number of Segments
gedit-change-case-plugin.html	162
gedit-create-new-file.html	123
gedit-document-statistics-plugin.html	157
gedit-edit-text.html	122
gedit-external-tools-plugin.html	216
gedit-file-browser-plugin.html	172
gedit-find.html	147
gedit-find-escapes.html	135
gedit-find-incremental.html	133
Total number of segments	7,198
Number of unique segments	1,093
Translated unique segments	57

You can find detailed statistic information in the file:
/home/arjun/gedit/omegat/project_stats.txt

Figure 3: OmegaT project files



file:///home/arjun/gedit/source/gedit-edit-text.html

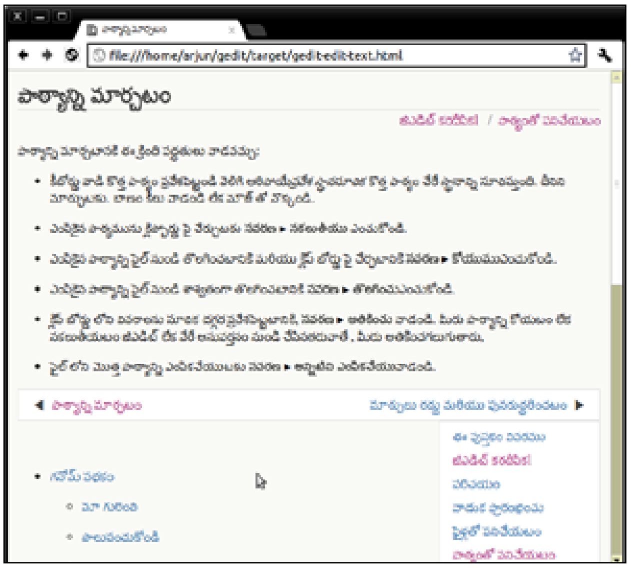
Editing Text

Gedit Manual / Working with Text

You can edit the text of a file in the following ways:

- Type new text from the keyboard. The blinking *insertion cursor* marks the point where new text appears. To change this, use the arrow keys on the keyboard or click with the mouse.
- To copy the selected text to the clipboard, choose Edit ▸ Copy.
- To delete the selected text from the file and move the selected text to the clipboard, choose Edit ▸ Cut.
- To permanently delete the selected text from the file, choose Edit ▸ Delete.
- To insert the contents of the clipboard at the cursor position, choose Edit ▸ Paste. You must cut or copy text before you can paste text into the file, either from gedit or another application.
- To select all of the text in a file, choose Edit ▸ Select All.

Figure 4: Gedit manual English sample page



file:///home/arjun/gedit/target/gedit-edit-text.html

పాఠ్యాన్ని మార్చుటం

[illegible]

Figure 5: Gedit manual Telugu sample page

left sub-window (Figure 1). Potential matches are displayed in the top right window, and glossary matches for words in the source string are shown in the bottom right. Using keyboard shortcuts or the mouse, the closest match can be selected and used as the translation. New target translation can be entered as well. As OmegaT is a Java utility, it has been noticed that the IBus-based input method does not work under Ubuntu; X keyboard map can be enabled to type the target language text in OmegaT.

OmegaT shows the progress statistics of the translation. All source strings except file names used for hyperlinking need to be translated. OmegaT does recognise mark-up tags. Tag verification needs to be done prior to producing the translated documents, to ensure that each opening tag has a corresponding closing tag. Figure 4 shows a sample page of the English Gedit manual and Figure 5 the same in Telugu.

In this article, we have looked at how a free computer TM tool like OmegaT can be used to translate user documentation. We will explore aspects of localisation style and other related issues in future articles. END

For more information

[1] OmegaT website: *https://www.omegat.rog/en/omegat.html*

By: Arjuna Rao Chavala

The author is an independent consultant in the areas of IT, program/engineering management and open source. He was a cofounder and first president of Wikimedia India. He initiated the IEEE-SA standardisation initiative in the software space in India and currently serves as the WG Chair for the global IEEE-SA project P1908.1, 'Virtual keyboard standard for Indic languages'. He can be reached through his website *arjunaraoc.blogspot.in* and Twitter ID *@arjunaraoc.*

Manual of
Style
and
Guides
THE DO'S AND DON'TS OF...
Typography
Design a
label-style
title
Words in a shape convey a pleasing blend of informality and permanence.

Localisation: Ensuring Consistency in Localisation Through Style Guides

This article in the series deals with the topic of style guides for localisation.

According to Wikipedia, "A *style guide* or *style manual* is a set of standards for the writing and design of documents, either for general use or for a specific publication, organisation or field. [It helps] provide uniformity in style and formatting across multiple documents."

I first learnt of style guides when my proposal for a conference research paper was accepted, and I was asked to send the complete paper as per their style guide. It consisted primarily of how to number sections, figures and references; the font sizes for headings of sections and sub-sections, as well as guidelines on other elements typical in a scientific publication.

In the context of localisation, we try to make a software application (or its documentation) developed for a *source* language appear equally natural for speakers of a *target* language. Languages differ in attributes, usage for communication, cultural elements, such as date formats, names of calendar elements and their short forms, and currency symbols. These need to be localised consistently. As the source language's elements and practices may have a different structure from the target language—or even be entirely absent in it—a style guide helps make the localisation consistent, even if many people participate in the localisation. The style guide deals with language and cultural attributes, terminology and quality assessment aspects.

In the following sections, I will focus on the difficulties common to Indian language localisers. I encourage localisers to use the references for more information on specific languages. I have used the FUEL and Microsoft style guides for Telugu for illustration and highlight the improvements needed.

Let's look at some language and cultural attributes.

Product and feature names

As product and feature names are trademarked, some style guides advise against localising them. For one, the Microsoft Telugu style guide states that product names should not be localised. Here is an example why:

```
The Microsoft Feedback Tool is unable to send feedback.
(+) Translation: Microsoft ఫీడ్‌బ్యాక్ ఉపకరణం ఫీడ్‌బ్యాక్‌ను పంపలేకపోతుంది.
```

From the viewpoint of user expectations, this is inappropriate. By transliterating *Microsoft* in Telugu as మైక్రోసాఫ్ట్,we do not violate any trademark rights. Actually, local laws demand that office signboards use the local language in addition to English. Using transliteration, the user interface looks more natural and friendly for a typical user

Technical acronyms
Technical acronyms such as OLE (Object Linking and Embedding), or RAM (Random Access Memory) are difficult to localise. When they are transliterated in Indian languages, they may be difficult to read—so for localising contexts where the user is supposed to have a bit of technical knowledge (like those who install operating systems) these acronyms can be left, as is. However, if they are to be presented to a lay user, then it is better that they are transliterated as individual characters or as a pronounceable word.

Culture-specific words
When we translate words like *India*, we prefer the more common Indic equivalent of *Bharat*, or its variations (variation 2, below) rather than a transliteration of India (variation 1, below). Similarly, when country names include the direction names, it is better to translate the direction name to the local language.

Example: From FUEL Telugu

English	Telugu (variation 1)	Telugu (variation 2)
India	ఇండియా	భారతదేశం
North America	నార్త్ అమెరికా	ఉత్తర అమెరికా

Keynames
Some keys have specific abbreviations like *Ctrl* and *Alt*, and most style guides suggest they be retained. I would suggest that as it is essential for everyone, transliterating the key name is more helpful.

Example: From FUEL Telugu

English	Telugu (variation 1)	Telugu (variation 2)
Both Ctrl keys together change layout.	రెండు కంట్రోల్ కీలు కలసి నమూనాను మార్చును.	రెండు Ctrl కీలుకలసి నమూనాను మార్చును

Foreign words ending with halanth

```
Meter- రోడ్, రోడ్డు
Road- మీటర్ , మీటరు
```

While the first form is correct from a phonetic perspective, the words are appended with a 'u' sound, which makes it more natural for Telugu speakers. However, if these are used in a context meant for professionals, the first form will be more appropriate.

Foreign words with multiple conjuncts for certain syllables

```
Software: సాఫ్ట్వేర్,  సాఫ్ట్‌వేర్, సాఫ్ట్‌వేరు
```

The first form is closer to English pronunciation, but is difficult for the user to understand. It is better to use ZWNJ (second case) to simplify the conjunct formation and improve readability, and also make it end with the 'u' sound (third option) to make it natural for Telugu people.

Short-cut keys
Different software use different ways (preceding with '_', or '&', or '~') to highlight short-cut keys. In English, these keys flow in the usual text. For Indic languages, as there are several input methods and some keys for vowel *matras* need the support of dummy glyphs, it is not possible to use local language short-cut keys. So, usually, the English short-cut keys are put at the end of the string, in parentheses. But these are still difficult to use, as this requires changing the input method, so I suggest that we dispense with the short-cut keys, as mouse and touch-based interaction is becoming the norm. However, it is an accessibility aid, so may require a longer discussion.

Terminology
There is no unified terminology for most Indian language localisation needs. FUEL is a project by open source software teams, and Microsoft has its own terminology. It is essential for a specialised language centre to build a common terminology that can be used by all.

Quality assessment
Currently, it requires other team members to spot deviations from the guidelines. Some localisation tools are able to flag elements like extra spaces. It would be very useful to have localisation tools check for conformance to style guides. END

For more information

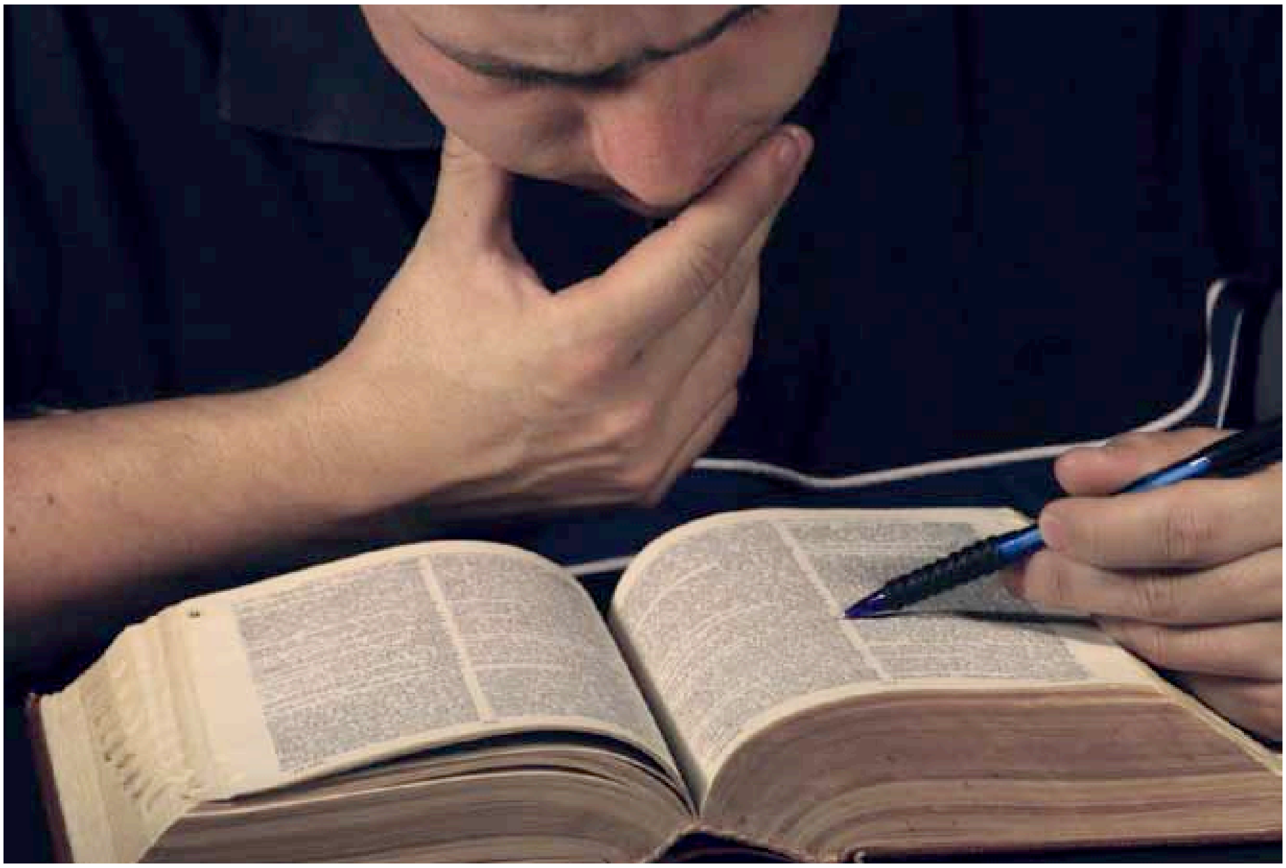
[1] FUEL style guides, *http://www.fuelproject.org/styleguide/index*
[2] Microsoft style guides, *http://www.microsoft.com/Language/en-US/StyleGuides.aspx*

By: Arjuna Rao Chavala

The author is chief consultant of Arc Alternatives. He serves as the WG Chair for IEEE-SA project P1908.1 "Virtual keyboard standard for Indic languages". He co-founded and served as the first president of Wikimedia India. He can be reached through his website *http://arcalter.com* or by email to *arjunaraoc@arcalter.com.*

Localisation: The Status of Education and Training

This article surveys the education and training opportunities in language services available in India and across the world.



Common Sense Advisory is an independent market research company serving the worldwide translation, localisation, interpreting, globalisation and internationalisation marketplace. Its estimate for outsourced language services in 2011 was US$ 29.885 billion. Europe (49.38 per cent) was the largest region, followed by North America (34.85 per cent) and Asia (12.88 per cent). There were around 26,104 language service providers (LSPs) with two or more employees. More than 95 per cent of LSPs have small-scale operations employing just a few people. The localisation market is projected to reach US$ 33.523 billion by the end of 2012, which means an annual growth rate of 12.17 per cent. Unfortunately, there are very few details about the Indian market in the open domain, though it is estimated at about US$ 0.5 billion in 2010, based on the IT enabled services market share.

People graduating with a diploma or degree in translation studies or linguistics courses join the LSPs. Some people who know more than one language work as freelancers. However, as localisation is a specialised area, there is a need for training on the theory and the tools. In the last decade, pre-conference tutorials as part of the annual Localisation World Conference, custom training from experienced professionals, and tool vendors offered the main training opportunities. A few universities and institutions have started offering offline and online courses and certifications in recent years. More details on such offerings are given below.

USA

In the USA, a Localisation Certification Program was launched in 2006 by California State University in partnership with professional associations. A new programme called 'Localisation Project Management Certification' for global website projects is also being offered.

Europe

In 2010, the Institute of Localisation Professionals headquartered in Dublin, Ireland launched a Certified Localisation Professional (CLP) programme comprising online and offline modules, and has trained a few hundred people. The programme is currently under revision. The Localisation Research Centre, part of the University of Limerick (Ireland) is offering a one-year M.Sc in Multilingual Computing and Localisation from the year 2011, in the distance education mode.

India

Several universities offer diploma and PG diploma courses in translation studies and linguistics. Students passing these courses found employment with government organisations and the media and served in translating administrative texts or literature/news. With the onset of liberalisation during the 1990s, the need for language professionals has grown rapidly, with India becoming the hotbed of IT outsourcing. During the last decade, the availability of computer operating systems and applications in Indian languages, and the rapid growth in the use of mobile phones and the Internet is further driving the market for language services. Most of you would have noticed the availability of Indian language user interfaces in ATMs, increased language options in Interactive Voice Response Systems and customer support centres of satellite television companies, besides the availability of English movies and television channels in Indian languages.

'Multilingual' magazine's December 2011 issue focused on opportunities in India. One reason mentioned for the small market for localisation is that the population that can afford to buy goods and services has a good knowledge of English, and would prefer to read material in English. However, the active involvement of the government through its Indian language initiative (Technology Development in Indian Languages); the focus of Google, Microsoft and other companies in their quest to reach a majority of the population; as well as the increased focus of the media and entertainment industries to serve Indians in their native languages, is ensuring that growth prospects in this domain are very attractive. Recent news reports mention the availability of enterprise applications in Hindi and other languages targeting the SME market in India.

The Indian government has started several initiatives to promote translation. The Institute of Translation Studies was set up in 2002 to promote offline and online education. The National Translation Mission Project was launched in 2008, with a target of translating knowledge texts into various Indian languages, and developing resources for improving translation in the country. It has published bilingual basic dictionaries of the most frequent words and phrases in several Indian languages. Punjab University started a Centre for Language Innovation in May 2012, with the objective of education and research in languages, to promote skills development in the related areas.

All the above initiatives were focused on translation. It is heartening to note that the Department of Electronics and Information Technology Working Group has proposed to set up a National Localisation Research & Resource Centre (NLRRC) during the 12th five-year Plan (2012-17). The objective is to spur localisation activities in India, with a focus on e-content in Indian languages, localisation of IT and non-IT products, localisation tools and platforms, standards development, as well as to promote entrepreneurship, incubation and Ph.D programmes.

Another important development is the launch of a professional association called the Indian Translators 3 Association in 2006, which is working with the government and others to organise the industry. The first international conference on the 'Role of Translation in Nation Building, Nationalism and Supra Nationalism' was held in New Delhi in 2010. The conference has become an annual event, and the association is also taking steps to introduce more professionalism into the industry.

In this brief article, we have reviewed the status of education and training in the broader area of language services. As can be seen, there is lot of growth potential in the language services market, and enterprising organisations and individuals can tap into this opportunity. END

References

[1] The Language Services Market 2012, Common Sense Advisory (Extract at *http://www.commonsenseadvisory.com/Portals/0/downloads/120531_QT_Top_100_LSPs.pdf*
[2] Localisation World Conference. *http://www.localizationworld.com/*
[3] Localisation Certification & Localisation Project Management Certification programme. *http://rce.csuchico.edu/localize/*
[4] The Institute of Localisation Professionals. *http://www.tilponline.net/*
[5] Multilingual magazine. December 2011 special issue on India. *http://multilingual.com/*
[6] Report of the Working Group on Information Technology Sector, Twelfth Five Year Plan (2012–17). *http://planningcommission.nic.in/aboutus/committee/wrkgrp12/cit/wgrep_dit.pdf*
[7] Indian Translators Association website *http://www.itaindia.org/*

By: Arjuna Rao Chavala

The author is chief consultant of Arc Alternatives, which works to catalyse the transformation of IT/engineering enterprises with a focus in the areas of IT, programme/engineering management and open source. He serves as the WG Chair for the IEEE-SA project P1908.1, 'Virtual keyboard standard for Indic languages'. He co-founded and served as the first president of Wikimedia India. He can be reached through his website *http://arcalter.com* or by email to *arjunaraoc@arcalter.com.*

The Status of Research in Localisation

In this 10th and concluding article in the series on localisation, the author take a closer look at the challenges for Indian language localisation and the current status of research in the field.

Ten years back, there were very few people who knew how to use Indian languages on the desktop. Now, we not only have millions of people who read content on the Internet in their native languages on desktops and smartphones, but also thousands of people who edit and contribute content. This has been possible due to the availability of computing devices with Indian language support, their falling prices, as well as the ability to access the Internet through various communication mediums.

We have seen a few early signs of the outcome of the research of the last decade, in the form of machine translation support for the Web. While five Indian languages are supported by Google's machine translation tools, the quality is still not up to the mark due to the complex nature of languages. Speech and touch interfaces have made their appearance, particularly on smartphones. The speech interface is now supported in limited domains, such as searching through the contacts list, or searching the Internet. However, Indian-accented English support needs to be improved.

As per the framework of the Centre for Next Generation Localisation, a specialised centre of excellence in Ireland, the challenges for localisation are volume, access (interaction mode) and personalisation. These three dimensions represent three axes, with most of the localisation work focused on high-volume content in corporate environments, with support for the traditional keyboard-and-screen mode of access, and limited support for personalisation in terms of language variations. The research challenge is to leverage various core technologies and frameworks to be able to instantly translate content and localise applications, duly considering the profile of the user. The next few paragraphs explore the status of localisation and the challenges faced in dealing with Indian languages.

a) Volume: The quantity of information on the Web is exploding due to its popularity as a medium of communication and interaction, and also the popularity of Web 2.0 platforms such as Twitter, Facebook, Google+, etc. The industry has tried to address this by defining and improving the process for localisation in corporate environments, as well as leveraging the crowd sourcing opportunity in social media environments.

The core component of localisation is the translation technology. For a long time, the route explored was rule-based translation research consisting of parsing of the source text, and using dictionaries and grammar rules to produce the translation. Subsequently, Statistical Machine Translation(SMT), based on training of the algorithms, with paired human-translated texts of source language text and destination language texts has become popular. Websites, translated at the click of a button for the dominant languages, have become feasible—though the quality of the translation could be inadequate for professional requirements.

Localisers can use the automated translation suggestions from SMT, when there is no proper match in translation memory to improve the translation. The resulting improved translation can be used to train the statistical machine translation system.

b) Access (interaction mode): The traditional access (interaction mode) method while working with computers is through a screen and a physical keyboard. We have seen the emergence of the touchscreen, which allows for virtual keyboards and alternate methods of input like writing on the screen or composing the input by rapid selection of letters from the virtual keyboard by tracing a finger from letter to letter. In addition, with the popularity of smartphones, voice input and output is becoming another key interaction mode.

Due to the small screen size of phones, there is potential for errors in inputting text. Dictionary-based approaches that prompt the user to pick a word from a limited choice have been helpful. Other technologies that have reached a level of maturity in English, but need further development for Indian languages, are spell-checkers and grammar checkers.

Speech technologies for text-to-speech and speech-to-text are critical for the voice mode of interaction. This works fine in a limited context like search or interactive customer support in English. The support for Indian languages is limited in text-to-speech, and barely exists for speech-to-text. And speaker independence and operating in noisy environments are global challenges.

Character recognition technology, which was developed to rapidly process huge volumes of data from physical books, supported by image processing and pattern recognition techniques, has matured for English, whereas current offerings for Indian languages are not adequate.

Handwriting recognition is another area of active research, as it allows for more natural user interfaces. Here again, the complex nature of most Indian scripts makes this a challenging research area.

c) Personalisation: Traditionally, localisation is coarse-grained in the sense of its focus on language and not much on its variation across countries and regions within a country. Personalisation refers to making information available as per the personal and information requirements of the user in a given context. This makes such information more valuable. If the user interface and other content can be made specific to a language as spoken in a particular region, the quality of localisation will become much better. This requires several resources, such as dictionaries at the dialect level and also a way to transform sentences from a standard language into its dialect forms.

Localisation tools

We have looked at the advances in tools from the basic text-editor kind of models to Web-based platforms in the previous articles. The tools have live interfaces to Translation Memory repositories, and support various project management tasks such as planning, tracking and work flow, as well as reporting mechanisms. Support for XML interoperability standards like XLIFF, TMX and TBX is also available. Several commercial business models based on the purchase and exchange of language resources have become common. Tools that allow Web localisation to be done directly on the displayed web page have appeared (e.g., Mozilla Pontoon). Further improvements to tools to manage the complexities of localisation as per user constraints, while leveraging Web services and crowd sourcing efficiently, is an active research area.

Future of Indian Language Technology Research

The Indian government's Department of Electronics and Information Technology (DeiTY) has an initiative called 'Technology Development in Indian languages' (TDIL). The objective is to popularise the support for Indian languages on computing platforms. It has been promoting work on machine translation systems—from English to Indian languages and from one Indian language to another, cross-lingual information access, and Optical Character Recognition and handwriting —through a consortium of academic institutions and research organisations for than a decade. Demo versions of products, along with relevant fonts and software for each language, have been developed and were made available through free physical CDs seven years back. The same are now available for download from its data centre website. However, all the offerings are only meant for non-commercial use. Redhat, Google, Microsoft and various small and medium enterprises have been pioneering their own initiatives to popularise Indic computing. Free and Open source groups have also worked tirelessly to improve support for Indian languages.
The involvement of all language computing stakeholders on a common platform in defining the strategic goals and assessing the outcomes, as well as releasing the results of basic research, tools and language related databases under unrestricted licenses will be a great step for rapid progress.

End note

It has been a great opportunity for me to introduce localisation and explore its various aspects, over the past year, through this magazine. I express my thanks to the *OSFY* editors and management for their support. I acknowledge and thank all the people and organisations who persevere passionately to make Indian languages on par with English in computing arena. END

References

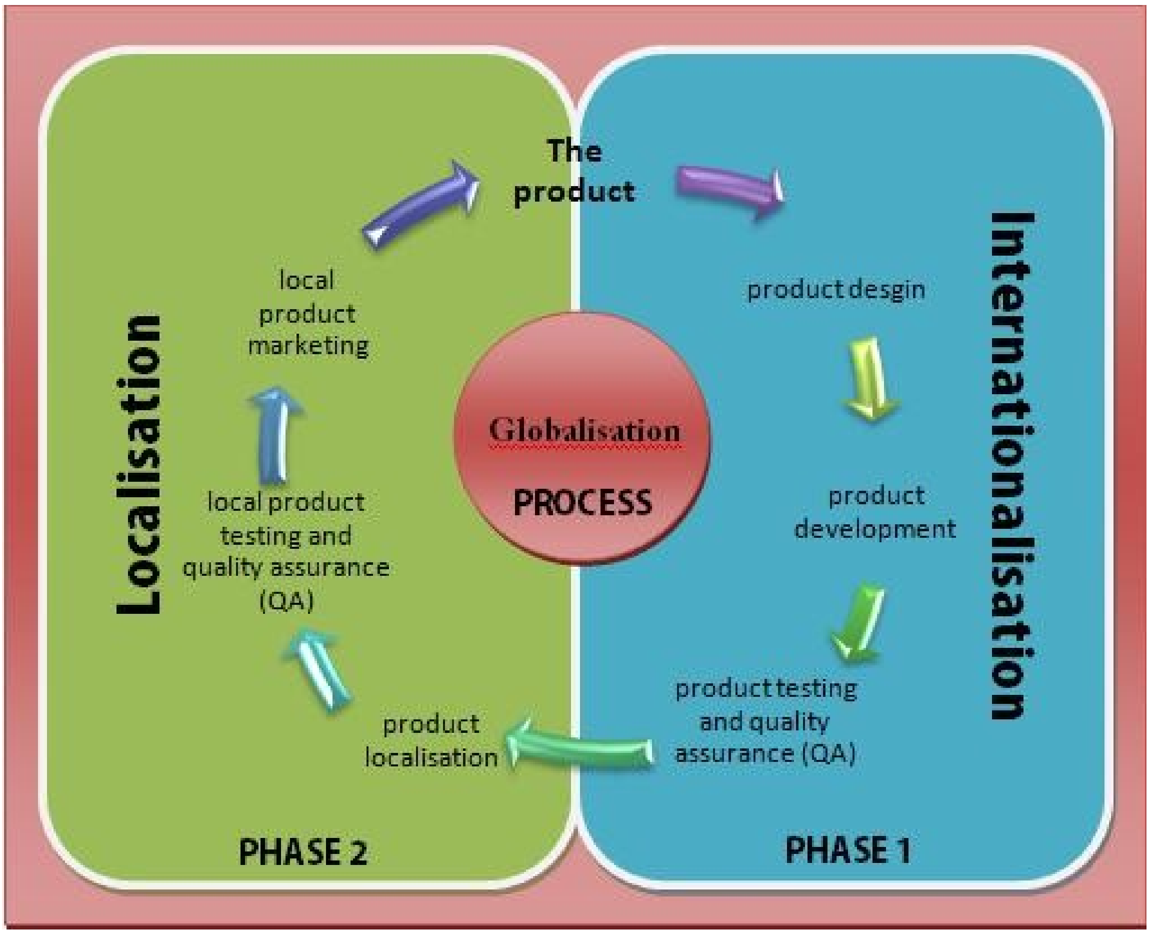
[1] Next Generation Localisation, Josef van Genabith, Localisation Focus, Vol. 8, Issue 1, *http://www.localisation.ie/resources/locfocus/vol8issue1.htm*

[2] Pontoon Introduction-Zbigniew Braniecki *http://diary.braniecki.net/2010/04/19/pontoon-introduction/*

[3] TDIL website *http://tdil.mit.gov.in/*

By: Arjuna Rao Chavala

The author is chief consultant of Arc Alternatives, which works to catalyse transformation of IT/engineering enterprises with a focus in the areas of IT, program/engineering management and open source. He co-founded Wikimedia India and served as its first president. He also serves as the WG Chair for the IEEE-SA project P1908.1–'Virtual keyboard standard for Indic languages'. He can be reached through his website *http://arcalter.com* or by email to *arjunaraoc@arcalter.com.*



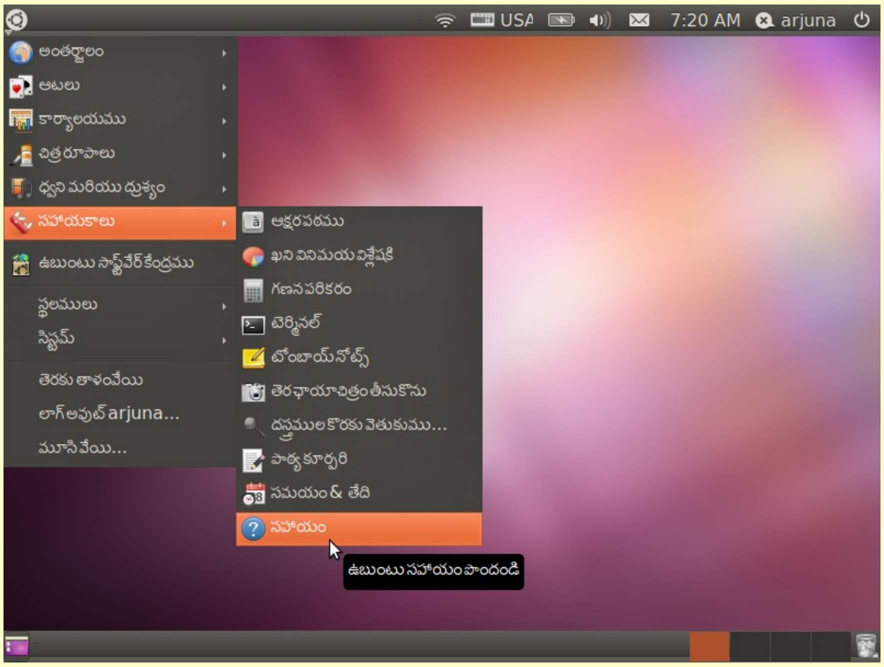
Globalisation Process (From Wikimedia Commons Author:Iat_ vicky)

Localisation

Arjuna Rao Chavala

Chief Consultant, Arc Alternatives

arjunaraoc@arcalter.com



Feedback from Readers:

Uday Mittal, LFY/OSFY Reader

I read your article named 'Localisation An Introduction' in Linux For You April 2012 and I found it very interesting. It was through this article that I came to know about Indian Linux Flavors like Rangoli and the importance of Localisation. I develop small applications and websites as hobby and would like to read the entire series of your articles in forthcoming issues of Linux For You.

Aditya Vinnakota, LFY/OSFY Reader

My name is Aditya vinnakota,working as web developer (opensourse). I read your localisation article in august month now I found (some articles in this blog). That was really really very nice and useful to me very effectively. I am very happy to meet you through this blog . Thanks for your knowledge share