Translate's Localization Guide

Release 0.9.0

Translate

Jun 26, 2020

Contents

CHAPTER 1

Localisation Guide

The general aim of this document is not to replace other well written works but to draw them together. So for instance the section on projects contains information that should help you get started and point you to the documents that are often hard to find. The section of translation should provide a general enough overview of common mistakes and pitfalls. We have found the localisation community very fragmented and hope that through this document we can bring people together and unify information that is out there but in many many different places. The one section that we feel is unique is the guide to developers – they make assumptions about localisation without fully understanding the implications, we complain but honestly there is not one place that can help give a developer and overview of what is needed from them, we hope that the developer section goes a long way to solving that issue.

1.1 Purpose

The purpose of this document is to provide one reference for localisers. You will find lots of information on localising and packaging on the web but not a single resource that can guide you. Most of the information is also domain specific ie it addresses KDE, Mozilla, etc. We hope that this is more general.

This document also goes beyond the technical aspects of localisation which seems to be the domain of other localisation documents. The document aims to help other localisation teams by consolidating the information of the Translate.org.za project and other localisation initiatives. So in this document we have added sections on selecting what to translate, budgeting, timing and more.

There are sections explaining how to use the Translate Toolkit and practical sections on common translation technical errors as well as ideas on how to translate or configure certain things like accelerators key, whether to translate acronyms, etc.

1.1.1 Duplication

We hope not to duplicate information and will point to other online resources if needed. But often we document in the process of creating our translation builds so there might be slight duplication.

1.2 Contributing

We welcome contributions – if you are a localiser then this is your document. Please think of this as a living document. If you are using part of it and find that it is out of date then please update it so that others don't have to go through your pain.

If you have a new section to contribute or some corrections then please create an account and make changes. If you feel it might be controversial then please email translate-devel@lists.sourceforge.net. If you have some criticism we also welcome it but if you accompany it with the changes then you are more likely to be listened to (its easy to complain, a little harder to contribute meaningfully)

We *credit* those who contribute whole sections and significant update.

1.3 Credits

These are the people and groups who have made this document possible:

1.3.1 Contributors

The Writers, section contributors:

- Translate.org.za – The foundation which made this possible in the first place.
- Dwayne Bailey – Capturing the experience of Translate.org.za to ensure that the pain for other localisers is considerably reduced.
- Javier SOLA – Documenting the usage of the tools, instructions aimed at newbies and at Windows users and the guide for OpenOffice.org localisation

Page contributors:

- Novica Nakov

The techies.

- David Fraser – Creation of mikiwiki to change this from a text only to a simple HTML document. Plus migration to the current wiki and various Pootle documentation.

Helpful input

The people whose input has helped shape the document.

- Tactical tech – Who seeded the idea of capturing this information.
- South African Department of Communications – Whose funding of Translate.org.za made the writing and capturing of this information possible.

1.3.2 Users

If you use and have found this document useful then please let us know.

- Translate.org.za
- Khmer OS

1.4 Copyright

The localisation guide is released under the Creative Commons Attribution-ShareAlike license.

1.5 Editing the Localisation Guide

We would like you to contribute the knowledge you have to make this guide more comprehensive. The main aim of the guide is to reduce the pain experienced by most localisers in finding the correct information on specific project and in contributing for the first time.

Please add your wisdom and help save others from experiencing your pain.

1.5.1 First things first

Please familiarise yourself with the guide so that you are aware if any of the content you want to add is already present. We don't want to duplicate information because if we do:

- The information become dated and disagrees across the different sections
- We're trying to make a guide that is as general and only specific when needed and then only where needed

1.5.2 What goes where

The wiki contains some distinct sections:

- The guides to the Translate Toolkit
- This guide – the localisers guide
- A glossary of localisation terms

Each of these make use of namespaces to distinguish sections from each other. Please make sure you are editing in the right place.

The localiser guide is broken down into the following broad sections:

- Running a localisation project in general
- Information specific to the major localisation projects
- Guides for translators that cut across all projects
- Hints for developers so that they can help localisers

Please place content in the relevant section. Please don't duplicate content but rather link to the relevant section in the other parts of the guide.

1.5.3 Headings

Generally each page has an H1 heading and all other headings are below that. Please make use of section headings for clarity (it also makes it possible for others to link to your sections). Also use high level headings starting from H1 in your document as this then creates the nice table of contents automatically.

1.5.4 Style

We are not that particular about the style of writing (well we haven't seen any awful stuff yet). But please do stay to the point and feel free to give personal examples and to use personal pronouns. Eg "we have found that". The guide text is quite formal but if you need to, then add humour as needed.

1.5.5 Spelling and Grammar

Firstly I can't spell! We also realise that some contributors are not native English speakers so don't worry too much about that – or ask someone on translat-devel@lists.sourceforge.net to check you spelling and grammar. The main idea is that it be understandable. Ask yourself, "Would this help me learn something, start something or start contributing?" If the answer is "Yes.", then your content is probably good enough.

1.5.6 Licensing

This guide is under a creative commons license. Please be aware of that before adding your content and be aware that copy and paste might violate someone elses copyright – rather write from scratch or paraphrase the documents you want to copy.

1.5.7 Credit

Everyone wants credit. Please add your name to the credits page.

1.5.8 Using Namespaces

The guide makes extensive use of namespaces, e.g. `translation/variables.html` refers to the variables page in the translation section of this guide.

1.5.9 Adding another language or translating

FIXME

Please make use of namespaces to separate your translations from the main document. Use your languages iso639 code as the namespace. For Zulu this page would be referred to as

```
zu:guide:editing
```

1.6 Localising the ANLoc manual

The African Network for Localisation (ANLoc) developed a training manual for FOSS localisation. You can find it here: http://www.africanlocalisation.net/foss-localisation-manual

If you are interested in localising this manual into your language, this page gives an outline of what is required. It is a big process, and takes a lot of time.

Although the manual can obviously be improved, we are very proud of the it, and want to make sure that translated versions are of high quality. Therefore we follow a long process with lots of review, and only aim for fully translated versions of the manual. Although a single person can probably do this, it is highly recommended that you get people to help you. It is most strongly advised for reviewing terminology and for reviewing the final document.

The manual is translated as a PO file, and converted to an ODF from which the PDF file is generated. The images are translated manually in a vector drawing program (Inkscape). Here is a rough outline of the process:

- We can add the necessary files on our Pootle server here: http://pootle.locamotion.org/projects/foss_manual/
- You will need to develop a terminology file, preferably in collaboration with other people who do FOSS l10n
- You can do translation on either Pootle or Virtaal, but need to upload to Pootle regularly for backup. I would recommend Virtaal for most of the translation. You will need to be familiar with both tools for the sake of the translation anyway.
- Then the translation can start. It is a lot of work (over 25000 words). You should know that you are ready for such a big amount of work :-)
- Review in the PO file for translation equivalence, quality checks, etc.
- We generate a translated document
- You review in the document, maybe more than once, start looking at things layout to do at the last stage (although we won't work on it at this stage).
- We can consider adding making some small changes specific to your language, such as changing or adding URLs, and even adding or removing some sections if the need for it is strongly evident.
- You translate the graphics
- Someone generates the translated graphics
- Hopefully we can get another review of the document
- We integrate the graphics and do another final review with the absolute last changes (manual layout changes, etc.)

1.6.1 Localisation brief: About the book

The book is part of ANLoc's mission of uplifting people in Africa through improvements in localisation. The message is by no means limited to Africa or Africans, but maybe it is good to keep this in mind as a start.

The typical readers include:

- Professional translators who don't know about localisation
- FOSS contributors who might want to learn more about localisation
- Existing localisers that might want to improve their skills. This is a useful book to give to new members in any team, and it should probably be accompanied by language and/or project specific guidelines for the team.

We didn't want the text to be too technical or demanding. We aimed at about 50 pages, and ended only slightly over that. So this is meant as a reasonably short introductory text, with links to useful external resources. The primary goal was for training events in Africa, but of course we are happy for anything it could be useful for.

We assume that the reader is mostly computer literate and can probably use a word processor and web browser comfortably already.

For the English version we assumed that most readers won't be native speakers of English, and might not have very good command of professional/technical English. For the future editions, we hope to do some readability studies. If you are translating into a language of wide use (with lots of second language speakers) let's aim for such a second language speaker, or at least keep them in mind.

All the examples are with English source text, and we propose that all the examples remain unchanged, including their translations. Please compare with the original PDF to see what the context is. Please don't try to translate the Northern Sotho examples :-)

It will be up to you to decide how this impacts your style, choice of words, and anything else you might think of. Feel free to share ideas or raise issues with us or on our mailing lists.

I'm sure you want to do this well, so don't be afraid to ask if anything is unclear, or there is a way in which we can help to make it better.

1.6.2 About the title

The book is not just about tips and tricks on the nitty-gritty of translation. It is not just a reference manual. We feel we incorporated useful instructions on how to think strategically and work on longer term goals, such as uniform style, team building, and ultimately, advancement of the language. We hope we made a convincing case for why localisation is important, and can be effective, even for currently marginalised languages (like most in Africa, the primary audience). The words "effect" and "effective" should therefore be kept in mind when reading the title.

So a longer, less elegant version of the title would be "How to change the world by doing localisation". You should fee free to translate the title freely to something that will work as a title, but please satisfy our curiosity and tell us what you are considering :-)

- *Purpose*
- *Contributing*
- *Credits*
- *Copyright*

If you would like to help expand this document then please take a quick look at our *editing guidelines*.

1.7 Start here

If you want some general introductory material about working on the localisation of Free and Open Source Software, maybe you want to have a look at this book produced by the African Network for Localisation:

http://www.africanlocalisation.net/foss-localisation-manual

It is available in English and other languages[1]. These wiki pages here tend to be more detailed, less structured and sometimes more technical.

1.8 Reading List

1.8.1 General Articles

These are as yet unclassified. Please reclassify if you see a need. We're not so concerned about quality here, but we are in the other areas.

- Localise! (July 14, 2006) – examines the Georgian localisation of FOSS and provides some interesting number on the coverage of mother-tongue speakers by Windows XP.

Some useful reading for the soon to be and established localisers. Please add good useful articles to these lists.

- *General Articles* – as yet unclassified but interesting

[1] You can *help localize the Anloc manual*

1.9 Managing a translation effort

1.9.1 Translate@thon

A translate@thon is a mass translation event (also called a localization sprint). Get a number of translators in one room and begin translating and you have a translate@thon. The general idea is that translation is simply a numbers game, get a large number of people translating and the task will be completed more quickly.

Have you hosted a translate@thon? Then please add your event to our *Translate@thon hall of fame*

Translate@thon Hall of Fame

If you have held a Translate@thon or Localisation Sprint then add yourself to the Hall of Fame.

Date	Where	Language(s)	Organisers	Software Target	Results	Outcome
Feb 2004	UCT, Cape Town, South Africa	Xhosa	PRAESA, Translate.org.za, WC Language Committee	Mozilla		:-)
Jan 2005	UCT, Cape Town, South Africa	Xhosa	PRAESA, Translate.org.za	Firefox 1.0	8000 words	=)
Mar 2005	UCT, Cape Town, South Africa	Xhosa	PRAESA, Translate.org.za	Firefox 1.0	8000 words	:-)
Mar 2005	DIT, Durban, South Africa	Zulu	DIT, Translate.org.za	Firefox 1.0	4000 words	:-)
May 2005	DIT, Durban, South Africa	Zulu	DIT, Translate.org.za	Firefox 1.0	4000 words	:-
April 2007	AITI, Accra, Ghana	Yoruba, Twi, Wolof, Hausa, Gun, Akan, Vai, Yemba, N'ko, Krio, Amharic	FOSSFA, Translate.org.za	Tuxpaint, Worpress subset	+- 600 per language	:-)
May 2007	Rhodes University, Grahamstown, South Africa	Xhosa	SANTED, Translate.org.za	Horde and IMP	10 000 words	:-)
Aug 2008	Makerere University, Kampala, Uganda	Luganda	Makerere, Rhodes, Translate.org.za	Firefox	? words	:-) 300 students

Comment on your event

Please do not add too many descriptive comments rather add comments that you feel would help other people run an event in the future. Also use this area to highlight why you rated your outcome as you did in the table above.

First Xhosa Translate@thon in Cape Town, South Africa

Neville Alexander, an ex-political prinsoner who spent time on Robin Island, started the event and noted that major events in history where often formulated late at night around a kitchen table. Fitting as the event was held in a basement lab.

We used Excel with CSV file. This proved awfull as we could not really integrate the work following the event and it took a long time to setup the computers. There were a lot of people 20+ and only one techie which was difficult.

Localisation training and Translate@thon in Accra, Ghana

We trained 16 people from mostly West Africa in many localisation issues and attempted Tuxpaint and a WordPress subset by translating Pootle. Pootle helped languages without keyboard layouts with the extra characters and provided a nice way of seeing the progress on the statistics page. This was the first time we translated from another language (Wolof, Yemba and N'ko localisers translated from French). Only the Amharic team had to use POEdit, since the special input method is needed. They could simply upload files without problems.

Expectations

To get the most our of your Translate@thon it is important that you understand what kind of outcomes you can expect and align your expectations accordingly. You will get different outcomes from small and large events.

- **Large event** – 40+ people. Most people will be new a translating and most probably unskilled. You will spend a lot of time explaining things, doing presentations, etc. Some people who are there might just be there for the social and will disturb others reducing their output. Expect low quality.
- **Small event** – 5 people. Usually the group is dedicated. The ratio of techies to translators is high so the quality goes up. The atmosphere is one of hard fun work as opposed to the carnival atmosphere at a large event.

So what can you expect?

Use a small event to get work done. Get high quality translations. Use a large event to expose people to localisation, to get media coverage and to promote Free Software. Large events are the places were you fish out the one or two dedicated translators who will carry on. At one large event we met with lecturers on a translation studies degree, localisation is now being worked into their degree.

Without a dedicated session or two devoted to review and quality assurance, expect there to be many problems with quality. Build in review sessions as part of your planning if you want to improve the quality and train people on the detail needed for high quality translation. Below more issues about quality is discussed.

Hosting partner

It might be good to involve a hosting partner. Traditionally we have chosen the language departments of Universities. They have access to good skills, computer labs and their own network of people.

Who should attend

You can take various strategies. Either throw the door wide open in which case you probably need to publicise the event slightly differently so that you can attract various people. Or make it a relatively closed affair with your hosting partner. For your first one it might be nice to make it closed as you can then ensure a quality audience. Remember that you have the chance to attract people who might not know anything about Free Software and who have their own networks that could tap into other potential translators. A mass event has issues about quality which we address below.

Venue

Your choice of venue should consider the following:

- Is it easy to get to
- Do they have enough computers

- Do they have Internet access if you need it

Remember to put lots of signage up on the day so that people can actually find the venue.

Preparation

Work with you partner to arrange the venue, people, etc. Make sure you have actually seen the computer lab and tested how things will work. If needed make contact with the lab assistant, manager, whatever and get their buy-in. They'll be a wonderful ally if things go wrong and you have brought them on-board.

It is also nice to arrange refreshment, prizes and talks. So plan for those and get sponsors if needed. And perhaps give a slot for people to talk on various aspect of Free Software and translation. Talks and refreshments give a nice break for people to network and get excited.

Ensure that you have a team of helpers that can assist newbies during translation, the number depends on the number of people expected. The ideal people are more computer focused then language focused. You need people who can quickly tell if something is a brand name, explain a computer term, describe what something does or how a term is used in computers. Having language specialists on hand is useful as they can direct the language aspect of the same word choice problems.

Choose a date so that the majority of people are available. Avoid the obvious like school holidays. Public holidays may actually be a good time to host the event unless they are family-oriented holidays or long weekends. If you want to attract students at University make it fall into their program, avoid exam times, holidays, etc.

What should you translate?

Ultimately the choice should align with broader goals for the language team. If this is the initial translation then you might want to translate a glossary, especially if you have access to language experts.

Another thing to consider is that if people have something to show for it afterwards then it is more rewarding. So translating the Mozilla web-browser and actually completing it so that people could take it home or download it shortly after the event.

A few days before

Remind people of the event and check the computers. Nothing is worse than no one coming or the computers not working. Check that you actually have access to the lab and its not closed for some obscure reason.

On the day

Arrange for yourself and helpers to arrive early. Check that the catering is OK. Recheck the computers. Welcome people.

Take it as it goes and adjust your program if it isn't working.

Program

This is the suggested program that we use.

Program	
09:00	Arrive tea, Register (hand out program, translation guidelines)
09:30	Start – intro talk (very basics of translation)
10:00	Translate short session
10:15	More talk covering more advanced topics
11:00	Translate
12:00	Lunch
13:00	Translate
14:00	Talk
14:15	Translate
15:00	Tea
15:30	Translate (hand out evaluation forms)
16:30	Close
16:45	Continue Translating if you want

This gives you 4 hours of translation time with none more than an hour long. Adjust as needed. If you have a mixture of new and experienced translators then it might be nice to arrange the venue so that experienced translators don't have to listen to any of the talks.

Give people a copy of the program and include the titles of the talks.

Event close

Don't forget to thank people. It is also good to get participants to fill out an evaluation form (example) as this allows people to give feedback. You can also use it to recruite people to the mailing lists and to help organise your next event.

Post translate@thon followup

Keep the energy going. Some ideas for this are to establish a mailing list. Send out copies of what was translated. Give prizes to those who did the most, had the least errors, made the funniest mistake. Share stories about errors that were funny.

Quality

How do you ensure quality of the work? These are people who have just started software translation and thus their work will be suspect, take that for granted. There are a few things you can do to increase quality.

- Ensure that people are well informed about common mistakes
- Have a document that gives a guideline to translators. The document should also identify things like variables, how to choose words, etc.
- Talk to new people before they start, have a quick 15 minute translation session then talk again to reiterate the issues.
- Never accept translations from a translate-athon until they have been reviewed by an established translator. If needed add them to the PO files but mark all of them fuzzy.
- Optionally only use professional language people: lecturers, translators
- Have computer people on hand to answer questions, or make technical and language people work in pairs.
- Encourage people to ask questions, regularly and often. This is important as some culture see asking questions as an indication of a lack of knowledge, be very concerned if nobody is asking questions.

- Plan for review, quality assurance and testing as part of the event schedule. The quality checks in Pootle is a great way to get everybody involved in reviewing the translations. The search feature might help to review terminology. Obviously getting the translated program running is good for reviewing things in context.
- For consistency at large events, it is probably worthwhile to prepare a terminology list before the event and install that in Pootle for terminology suggestions. This will at least eliminate certain types of inconsistencies. Translation memory is another way to help, although that might be harder to setup, depending on the administration skills available.

Checklist

- Planning
  - Found partner
  - Determine appropriate day
  - Create mailing list if needed
  - Arrange prizes
- Lab
  - Lab suitable
  - Meet with lab supervisor
  - Book lab
- Venue
  - Place for tea
  - Tables and chairs for tea arranged if needed
- Few days before
  - Check lab booking
  - Reminder to participants
  - Contact assistants, technical and language to confirm
  - Plan and create program
  - Print evaluation and translation guides
- Day before
  - Visit lab
  - Check on caterers
  - Arrange tea area
  - Arrange registration area
  - Meet with assistants to discuss their role and your expectations
- On the day
  - Setup for tea or check on caterers
  - Bring all forms (registration, program, guidelines)
  - Force your assistants to mingle especially if they are already project participants
  - Thank people

    - Hand out prizes

- Followup
    - Add people to mailing list
    - Email thanks

1.9.2 Setting Project Objectives

What are the aims of your project? Without that defined you are bound to wander aimlessly and might be surprised when what you thought were your objectives are not being met.

Some misguided objectives

It is always useful to look at some poorly formulated objectives and work from there. So here are some examples:

Disconnected from reality

"Our goal is to ensure that people who speak Eastern Blowfish can use computers in their mother tongue". The Eastern Blowfish team then decides to localise GNOME. However, the reality in West Smedburg is that people use Windows and end-users have not adopted Linux in any great numbers. The result is an effort to localise which will probably be successful at localisation but not at impacting the lives of the stated users as they are all still battling away in English on Windows.

A real world example. The Translate.org.za project initially selected the *KDE* as its first localisation project. The goal was a little more defined in that we hoped to impact users at the same time as promoting Linux. When we readjusted our objectives we realised that if we wanted to impact language speakers then we would have to be localising cross-platform projects such as *OpenOffice.org* and *Mozilla Product Localisation*. Thus our objectives have shifted.

Another example is that of a localiser who I met who reiterated the Eastern Blowfish example but for a real language, his mother tongue. However, on challenging him it turned out he was his family's computer guru. He wanted them to use computers in their mother tongue. So really his objective was: "I want to localise so that my mother can email using GNOME in her mother tongue". This is a perfectly valid goal – just don't confuse it by saying your doing it for everyone.

"I translated my Linux distribution's installer because that is where people start." The truth is that most computer users don't install their own operating systems, and that such installation is only done on occasion. Translating software that people use daily will have much more of an impact.

1.9.3 The Golden Rules

These are the Golden Rules of localization direction setting initially used by Translate.org.za. They are born out of the goal of wanting to have the greatest potential impact on language speakers with localised Free Software.

There are three rules, software that we translate must be:

- End-user Focused
- Free Software
- Cross-Platform

With these basic rules in mind you can more intelligently select your software targets.

End-user Focused

The person who will most benefit from localized software is the desktop bound end-user. It is not the sysadmin or programmer. These are the people most likely to have less command of English while sysadmins have probably in the course of their work had to come to terms with English on the computers that they use.

This also means that you need to examine what type of software is generally used by an end-user. This would include: office suites, email programs and web-browsers, instant messaging and even games.

Free Software

Again if you want to make the most impact on the most people you need to remove barriers. Free Software removes two barriers:

- being Free is the very reason why you can localize the software with relative ease (No NDAs, etc), and
- your localised software is then available to a broad community with easy sharing and community based sharing dramatically reducing the cost of acquisition.

Cross-platform

The reality is that most computer users are using Microsoft Windows. Until that changes a heavy focus should be on cross-platform products. Thus you can provide a solution to Windows user while at the same time providing an avenue for them to move to a Free operating system.

It is much easier to provide a localised office suite solution that does not require a complete retooling of the users operating environment.

1.9.4 How Translate.org.za applied the Golden Rules

These are the Golden Rules applied to the target selection at Translate.org.za:

- OpenOffice.org
- Mozilla
- A linux desktop
- Installers

Here is the explanation for this choice. OpenOffice.org is a leading Free Software, End-user focused and cross-platform piece of software and so meets all of the 3 requirements. However, so is Mozilla (and its offspring, Thunderbird and Firefox). We place OpenOffice.org before Mozilla because when using OpenOffice.org you are immersed in the language because you are probably editing an Afrikaans document while using an Afrikaans interface. While a web-browser gives you an Afrikaans window frame onto an English Internet. the argument does not apply as well to an email reader such as Thunderbird in that you are creating content in an email program. This is when you need to make a call.

Mozilla is localized before we look at any Linux desktops simply because people use Windows and Mozilla fills a gap in the needs of the end-user.

Translate.org.za has not made any firm decisions about desktops. We have localized both KDE and GNOME in the past and continue to do various depending on conditions and funding. The project sees this as an important step in creating a fully localised Free Software environment but is at a lower priority while there are still low hanging fruit in the cross-platform area.

We relagate Linux installer and distribution specific localisation to a lower rung. Distribution specific configuration programs are problematic in that they have limited impact to the distribution as apposed to a whole sea of Free Software users. But they are important for a seamless end-user experience. Some of them are also only used once, for example during software installation making localisation of them wasteful of resources.

Not listed are the localisation of kernel messages and command line tools. This is moving far beyond the realm of the average end-user and until a growth is seen in usage in this area they will remain off our radar.

1.9.5 What to do as a single person or a small team

The programs mentioned above are very big, and you might not have enough time to attempt such large projects yet. It is also good to start with something smaller, which will give you the satisfaction of completing a project, and give you something with which to encourage others to help you.

Here are some ideas of projects to look into:

- Tuxpaint – A drawing program for children. This is a very small translation (less than 1000 words) and the program is not dependent on any libraries or GUI toolkits that need to be translated.
- Pootle – A online system for translation and translation management. This is not such a high priority program to translate, but it is easy to do it, and provides lots of help. Translating Pootle can therefore also help to train new translators in your team.
- GCompris – Educational software for children. This is quite a large translation (more than 10 thousand words), but it is fairly easy to prioritise the effort either by targeting specific activities, or by doing the *One, two, short, long*. GCompris is not dependent on any libraries or GUI toolkits that need to be translated.
- xdg-user-dirs – Translations for the common directories on Linux type systems (Documents, Desktop, Videos, Music, etc.) It is a very small translation (less than 50 words) and the effects should be visible in most "Open" and "Save" dialog boxes on Linux systems. Also see the project page.
- GTK+ (look for the UI translations, not the properties). This is the GUI toolkit used by all programs in GNOME, as well as many cross platform programs like GIMP, Pidgin, Abiword, Dia, GnuCash and Gnumeric. Although all the UI translations in GTK+ are more than 3000 words, you can start by doing only the "stock" translations (look for "stock" in the #: comments) which is just over 100 words. These "stock" messages are things like "OK", "Cancel", "Open", "Save", "Print", "Warning", "Undo", "Preferences", etc. These are used by many applications, so translating them will give an initial boost to the localised feel of all your programs using GTK+.
- Pidgin – A chat program for several protocols. This translation is quite big (more than 15 000 words), but it is easy to avoid translations that deal with certain obscure protocols (check the #: comments). Pidgin uses GTK+, so at least a partial translation of GTK+ will be needed to have Pidgin 100% translated.
- Audacity – An audio editor. This translation is not that big, but it contains a lot of technical terms that might make it a bit more difficult. Audacity depends on the wxWidgets toolkit. For Linux this means that GTK+ should be translated as mentioned above. For Windows, wxWidgets need to be translated specifically, but it should also be possible to limit your work to the most important messages.

1.9.6 Profiling through message marking

If you can quickly determine where a string in the user interface originates from then you can determine what strings and packages are more important to translate.

To achieve this you need to be able to visually identify the strings origin. This is achieved by using podebug from the Translate Toolkit.

Advantages

It is simple to use and can also be used for correcting errors in the translations. By simply running an application or computer environment in the same way as your target audience you can quickly determine what parts of the system are being called and therefore what aspects you will want to translate.

Using the podebug method

podebug is simple to use and will take a directory of translated PO files and create a directory of tagged PO files:

```
podebug -f "[%cb] " -i af -o af-debug
```

Once the debug PO files are created then the normal build process is followed for that project.

Existing profiling builds

The following are builds that have already been profiled. Simply download, install and run the program as normal, making notes of the source files identified in the various screens.

- OpenOffice.org

1.9.7 Using profiling to aid direction setting

In computer programming the term profiling is used to indicate when a program is analysed to see where it spends most of its time. The theory being that that is where you should put in effort to optimise the code and thus improve speed and responsiveness.

A similar concept can be applied to translation. If you translate only the messages that people actually see then you can optimise the effort that you expend on translation.

It also helps focus the translation effort onto programs that are actually being used, not those that you guess might be being used.

How profiling works on Linux

The gettext libraries are called by a program with a request for the translation of a given string. Gettext looks up the string and returns the translation to the program.

When profiling this call to gettext is intercepted and the message requested is output to a file. This file is then later processed and contains all strings that should be translated.

Problems with profiling

- Many **UI design tools** and newer UI code calls all strings when initialising the UI. That means that you see many messages that in fact were never seen by the user.
- **Spurious files**. Many GUI program call command line tools to retrieve data. When the command line tool is called those messages are logged but were never seen by the user.
- Profiling depends on the **usage profile of the user**. An end-user and a system administrator will use different tools. To work best you need a group of people using your target applications.

Solutions

- The UI design tools would require changes to only call strings for windows that are visible. But this would only happen if translators can indicate a strong case for the fact that they use and find profiling effective.
- You can circumvent the calling of spurious command line tools by GUI tools by using:

  ```
  export LC_ALL=C
  ```

1.9.8 Using profiling

Gettext

This is the preferred method but it only creates one file. The gettext manual has a good description of the process.

The simplest first steps are repeated here for clarity:

```
$ LD_PRELOAD=/usr/local/lib/preloadable_libintl.so
$ export LD_PRELOAD
$ GETTEXT_LOG_UNTRANSLATED=$HOME/gettextlogused
$ export GETTEXT_LOG_UNTRANSLATED
```

Use the application, unset the variables and then:

```
$ msguniq $HOME/gettextlogused > missing.po
```

To remove duplicates. Various domains for each of the applications logged will be available. To sort and extract the domains that you wish to edit you should read the manual pages.

Gettextlog

- The Gettextlog tool website
- An RPM is also available

Once installed

```
export LD_PRELOAD=/file/to/gettextlog.so
```

You can place this in your `.bash_profile` to log continuously or in `/etc/profile.d/` add a `gettextlog.sh` script to initialise LD_PRELOAD

You can run it against individual programs with the `run-with-gettextlog` program

Your profiles will be output to:

```
$HOME/gettextlog/$PROGRAM.po
```

1.9.9 One, two, short, long

How do you translate most quickly for the biggest impact? If you look at a GUI application you will notice that most strings are relatively short. Menu items are between one and three words long. Error messages are more descriptive at perhaps 5-10 words. So in general the items that users see in normal operation are relatively short.

Therefore, if you can initially focus your translation on the short strings then you can make a quick visual impact. Afterwards you can follow up with longer and longer strings until the application is fully translated.

Shortcomings

The problem with this method is that you lose context in that you are not sure in what situation the single word string was used. Did “Manual” mean “configure manually” or the “operations manual”.

Using pogrep to select messages

Use pogrep to extract messages that are a certain number of words long:

```
pogrep --search=msgid -e '^\w+(\s+\w+){0,3}$' -i templates -o short-words
```

This will extract between 1 and 4 words from *templates* and place them in *short-words*. Once these strings have been translated you would use pomerge to combine them with the full PO file.

Use the following regex’s to extract specific word counts:

Words	Regex
1	`"^\w+$"`
2	`"^\w+\s+\w+$"`
3	`"^\w+(\s+\w+){2}$"`
4	`"^\w+(\s+\w+){4}$"`
1-4	`"^\w+(\s+\w+){0,4}$"`

Another way might be to look at all messages shorter than a certain number of characters:

```
pogrep --search=msgid -e '^.*{,30}$' -i complete -o short
```

This will extract messages that are up to 30 characters long, but not longer than that from *complete* and place them in *short*. Once these strings have been translated you would use pomerge to combine them with the complete PO file(s). This way it is sometimes possible to translate more than 60% of the messages by translating less than 40% of the words. Use pocount to see how much work has been isolated by pogrep – this will help you to plan your time.

Example of the potential impact

As an example Mozilla 1.7.3 contains 32217 words for translation. However, breaking it down into the number of words in short strings you get the following:

Words	Word Count
1	1853
2	1774
3	1371
4	796

Or in total 5794 words, which is 17% of the original total.

1.9.10 Direction setting based on the installation and bootup process

This document looks at the whole Linux installation and bootup process with these main sections:

- Installation
- Booting

- Init 5 (GUI)
- Init 3 (Text)

The bootup and installation process are very visual and give a user the first impression of the level of localisation in their software. If we can give them a good impression with less work then it becomes easier to create a fully translated Linux distribution.

The idea is to layout the dependencies and try place translatable components in some relative order in the process. This is done so that we for instance can place the language locale file at the correct point in the bootup process. And thus at the correct point at which it becomes important to a localisation project.

FIXME This document is based on Fedora but probably applies to any Linux distribution.

Installation

Stage	Messages	Description
CD boot messages	(unknown)	
rhpl	134	lists mice keyboards and exceptions might be needed in bootup as it mentions failed mouse detection and not being able to do graphical install
Comps	158	package groups and some descriptions used by anaconda I presume
Anaconda installer	1473	
Anaconda online help	263	
Firstboot	73	

Booting

Stage	Messages	Description
Grub	?	not sure how to localise (localisation not implemented only scheduled post 1.0 release)
kernel		The Linux kernel messages are localisable
Locale files	?	needed by some initscripts to display date
Initscripts	29	only 'sysinit' and some 'init.d/funtions'
rhgb	29	
Kudzu	70	

END OF BOOT PROCESS the next stage depends on the run level, most end-users will never see run level 3

Init 3

Stage	Messages	Description
/bin/login		found in util-linux – should try and extract only /bin/login strings

Init 5

Stage	Messages	Description
GDM	695	lots of bumf but hard to separate it out
Wallpaper		
Fedora	5	artwork

Post Booting

The booting process will take a computer to the place where the user is required to login. This section looks at what items are seen just after the user logs in.

Init 5

- Wallpaper (if different from login wallpaper)
- Fedora – menu's
- GNOME / KDE
  - Kicker
  - K Menu bar general
  - Clock
  - Desktop

Init 3

- /etc/issue not really possible to change easily

Config Tools

Once logged in a user will interact with the standard Free and Open Source applications that are usually localised outside of the distribution community. However each distribution has its own set of configuration tools that need localisation.

FIXME Create some category of which configuration tools are more important to localise then others.

On Fedora all the user centric and hardware related system-config tools constiture 7691 words of translation.

1.9.11 Existing Glossary Resources

The following are existing glossary resources. Please first check that you are indeed allowed to use them.

For Free and Open Source Software, the single most useful resource is probably at http://open-tran.eu/ which allows you to view translations from many of the major FOSS applications. Support for Open-Tran.eu is integrated into some translation tools, such as Virtaal.

The FUEL project tries to coordinate terminology development amongst Indic languages.

Glossaries

- Novell
- Microsoft
- Apple (2)
- Microsoft "community" Glossaries
- Skype

Domain Specific

...

1.9.12 Creating Glossaries

Clunky method to create glossaries

Note: A newer tool called poterminology is now part of the Translate Toolkit and should work better for all users. This page is only left for reference.

This old method is for Windows users, but the principles can be applied in Linux as well. It is for creating a subject specific glossary in a specific software translation project. The idea is to get a list of commonly used jargon from the project that can be translated beforehand.

1. Extract the source text

Take the PO file and do po2csv on it:

```
po2csv file.po file.csv
```

Open the CSV file in Excel or Calc.

Copy the "original" (or "source") column into a text editor, like Babelpad, and save the file as text.txt.

2. Extract commonly occurring words

Use ExtPhr32 to extract the most used terms from it:

- Do not select a stoplist (click "Cancel" when asked)
- Go *Options* → *Minimum occurrences*, and select "1"
- Go *Options* → *Maximum words in phrase*, and select "1"
- Go *File -> Extract from*, and select the file with source text
- In ExtPhr, change the "Minimum occurrences" higher and higher until you have about 200 terms (or, about 5 occurrences should do it).
- Save the result as 5ormore.txt.

3. Create general common words list

Create a file containing the 3000 most used words in English. I used Ogden's 850 words, plus the 3000 words, adjectives, adverbs and pronouns from here, but you could use Kevin's lists as well (actually, that would be better because then you could legally distribute it).

4. Remove general words from extracted common words list

Use Kastrul to spellcheck the "5ormore.txt":

- Go *File* → *Open dictionary*, and select your 3000 most common words.
- Go *File* → *Check file*, and select 5ormore.txt
- Copy the result, and save it as glossary.txt.

That is your glossary. Remember to change it all to lowercase, and to ensure that dud words are removed.

When localizing computer software, especially into languages where IT terminology is not available, new IT terms need to be created. If a glossary of computer terms does not exist in that language, then one needs to be created as these terms stand as a 'backbone' in the translation work.

Glossaries can also be created for specific projects (Here is a *clunky way of doing it*).

Resources

- KiPot
- How to make a simple glossary

How glossaries are created

The process of creating new terms involves joint work of linguistic and computer science experts. The development of new IT terms requires that the linguistic expert fully understands the meaning of the IT term and its context before trying to match or extend the meaning of a word in the destination language.

In the words of Alberto of the KiLinux team: "After completing our first Swahili IT Glossary of 700 terms we understood very quickly two things: the first is that the glossary of 700 terms requires to be extended constantly and second the extended glossary requires a constant and fluent communication between our technical team and Swahili linguistic experts. Oral communication was never an option since answers were required promtly. Email was not suitalbe either, since we wanted all participants to have the possibility to take part in the deveopment of all new tems, which would lead to an enormous amount of mails sent back and fourth between the projects participants. We needed an online communication tool that would facilitate the communication, and therefore, KiPot was developed."

Hacking a glossary from existing translation

One option to create your first cut glossary is to use existing translations and translatable applications.

The first step is to create a list of potential glossary words.

```
$ pogrep --search=msgid -e '^\w+(\s+\w+){0,1}$' -i templates -o short-words
$ rename .pot .po `find short-words -name "*.pot"`
$ pocompendium glossary.pot -d short-words/
```

Optional: edit the POT file to remove words that shouldn't be in the glossary. See pogrep and *One, two, short, long* for more information on the regex used to extract the words.

Now the glossary POT file is eady for populating. The next step is to take your existing translations and populate the POT file to create your first cut language glossary.

```
$ pocompendium compedium-XX.pot -d XX/ # Create a compendium for the XX language
$ cp glossary.pot glossary.po
$ msgmerge --compendium=compendium-XX.po --update glossary-XX.po glossary.pot
$ po2csv glossary-XX.po glossary-XX.csv
```

Your glossary is now in CSV format in the file glossary-XX.csv

It may be useful to provide translators with a web-based archive of reference documents that pertain to their translation. This would help them do highly focused searches in online documentation etc.

1.9.13 Create a custom Google search engine

Google offers a service whereby anyone can create a specialised search engine by limiting the pages and web sites that are indexed. The service has several additional features but for our purposes the most basic features would suffice.

- http://www.google.com/coop/cse/
- You must have a Google account (free registration)
- You can change most aspects of it later, so don't worry about putting in the wrong information.
- You can specify just one or several pages/sites to index, and you can specify them in interesting ways (eg only pages that contain the letters "about"). You can also exclude pages by a pattern.

Unfortunately, the URL of the search engine is a bit difficult to remember, but you can easily link to it from any page.

Here is a test search engine that will search only pages of this wiki, as well as pages from the Wikipedia that contain the letters "transl" in their URL:

- Translate Wiki Search Engine (try doing a search for "machine").

1.9.14 Adding a custom search in Opera

How to create a custom search in Opera 9.5

- Go to a page that contains the search box.
- Right-click the search box and select "Create search".
- Type in a name and a shortcut for the search (e.g. "Art of Illusion" and "aoi").
- Done!

Methods to use:

- Select text, then right-click the selection and use *Search with...* → *Art of Illusion*.
- Double-click a word (it will be selected), then use *Search with...* → *Art of Illusion*.
- Open a new tab `Ctrl+T`, then type "aoi searchword" and `Enter` to search for "searchword".

1.9.15 Adding a custom search in Firefox

How to create a custom search in Firefox 2

- Go to a page that contains the search box.
- Right-click the search box and select "Add keyword for this search".
- The caption will read "Add bookmark", but don't worry, you're actually adding a search.
- Type in a name and a shortcut for the search (e.g. "Art of Illusion" and "aoi").
- Done!

Methods to use:

- Open a new tab `Ctrl+T`, then type "aoi searchword" and `Enter` to search for "searchword".

1.9.16 Skill Requirements for Translators

This skill list is of course probably idealistic as you have no real control over the skills that volunteers will bring to your project.

Essential attributes summary

- Mother tongue speaker
- Passionate about their language
- Computer experience

Nice to have:

- Graduate qualification: Computer Science, linguistics, language

Essential skills explained

Mother tongue speaker

Many people say they are proficient in multiple languages. Some people are, most people aren't. The problem with people with multi-language skills is that they often do not have access to a deep understanding of either language. And a deep understanding is what you need if you want to find words that can be reused in a different way in a computer translation. So treat the polyglot with caution until proved otherwise.

Passionate

This may seem redundant but a person who is passionate about their language is much less likely to say well there isn't a word for "Proxy Server" and is more likely to understand the need for a good and extensive translation.

Computer experience

Someone who has no computer experience will not have the best understanding of words used differently in the computer world – words such as 'execute' and 'run'. But a BSc is also not needed. You need someone who can comfortably use a computer they can always lookup difficult words in online computer dictionaries.

Graduate qualifications

This is a nice skill to have especially when defining terminology that requires a deep understanding of language, etimology and computer usage and definitions.

1.9.17 Budgetting for Paid Translators

Although we encourage the volunteer model there are projects that are funded and there are projects that receive funds from time to time to do a specific translation. This gives a rough outline of how to cost such a translation.

Calculation

```
Words = untranslated msgid words + fuzzy msgid words
Cost = Words to translate * Rate per word
Days = Words to translate / Words per days
```

Rule of thumb

```
Weeks = Strings to translate / 2000
Days = Strings to translate / 400 or 500
  ( Use 400 for large project 500 for a small project. This then evens out
  lost days or slow progress due to complicated section.)

Strings = Words / 3.5
  (This varies but is helpful)
```

Calculating Costs

FIXME convert these numbers to dollars or other 'international' currency

Translators will quote based on the number of source words:

```
R400 / 1000
R40 / 100

which calculates to R0.40 per words
```

Note: these are just for illustration purposes

Using pocount determine the number of source words. So this is the number of words in untranslated strings plus the number of words in fuzzy strings.

```
Words = untranslated + fuzzy
Words = 40 000 + 5 000
```

Note: make sure you count the msgid or source words not the translations.

You final costing would then be:

```
Cost = 45 000 * 0.4
     = R18 000
```

Calculating Time

Assuming that you need to translate 45 000 words do the following. Either,

- Find out how many words the translator can translate per day

```
1300 words per day

Days = 45 000 / 1300
     = 35 days
     = 7 weeks
```

- Convert to the rule of thumb guides above

```
Strings = 45 000 / 3.5
        = 12 857
       +- 13 000

Days = 13 000 / 400
     = 33

Weeks = 13 000 / 2000
      = 6.5 weeks
```

- Running a *Translate@thon*
- Setting direction
  - Philosophical
    * *Creating project objectives*
    * *Golden rules*
  - Automated methods
    * *Message marking*
    * *Profiling*
  - Practical
    * *1, 2, short, long*
    * *Let the bootup process be your guide*
- Glossaries
  - *Existing Glossaries*
  - *Creating Glossaries*
  - *Create a custom Google search engine*
- Translators
  - *Skills required*
- *Budgetting for Paid Translators*

1.10 Project specific information

1.10.1 GNOME

GNOME is a feature rich Linux graphical desktop environment. The project has quite an established localisation initiative.

Since this page has been created, many things in GNOME has changed, and they have good documentation and infrastructure to help translators.

The most important documentation for translating GNOME is therefore at the GNOME website: http://wiki.gnome.org/TranslationProject/

The rest of this document is kept in place for the time being, but a lot of it is now out of date and not 100% accurate any more.

Essential Reading

Please familiarise yourself with these before proceeding:

- GNOME Translation Project
- Localisation Guide
- Join the GNOME Translation Project

Useful URLs

- Status Page

1.10.2 Translating

This is a roughly sequential outline of the steps you need to take to translate GNOME into your language.

Joining the mailing list

Subscribe to the gnome-i18n mailing list

GNOME Glossary

Start with GNOME Glossary to create consistency across future GNOME localisations.

Start with the latest versions in of GnomeGlossary.csv in CSV format and convert to PO using the csv-to-pot.sh script. The layout is slightly different from the layout created by po2csv so you cannot convert the created POT file to a Translate Toolkit style CSV file.

If you need to translate using CSV instead of PO then rather edit the GnomeGlossary.csv and manipulate it into a PO file later.

However, please note that the glossary itself is several years old and **not maintained** anymore (confirmed by last maintainer), so don't depend heavily on it. Instead, use it as a reference, and build all necessary glossary term as needed.

What files first

Advice from Christian Rose on the GNOME team.

Application	Strings	Description
gtk+	1126	toolkit, very largish, but many messages are developer-oriented and can safely be ignored to begin with, but some few messages here are very visible to the end user
libgnomeui	305	many user-visible menus and stuff
gnome-mime-data	350	user-visible file type desc. etc
libbonoboui	96	
gnome-vfs	80	file size formatting etc
yelp	71	help browser
gedit	640	text editor
nautilus	1449	file manager, also very largish, but not all messages here are immediately visible to the end user, even though many are
gnome-desktop	88	
gnome-panel	587	
gnome-session	103	
gnome-control-center	649	
gdm2	629	login manager
eog	170	image viewer

... and then the rest in desktop + developer-libs.

FIXME use podebug to get a better targeting on these files.

Committing translations

Translations can be committed directly from Damned lies in most cases. In the rare occasions where this won't work, you can ask on the gnome-i18n mailing list and someone should be able to do it for you. If you really need a git account, you should read the git related pages listed on the wiki

Targeting a release

GNOME follows a regular 6 monthly development cycle with even numbered stable releases and odd number development releases. The release schedule will help you decide which release to target.

If your team is moving quickly it might be good to target a stable minor release. This will also be the platform that most users will be on. It also presents the chance to have multiple releases as you move through each minor release. Otherwise, it would be more realistic to spend translators' effort on next major stable release.

Translation Status Page

The translation status page keeps up to date statistics on the progress of each language, your language should appear as soon as your first file is committed to CVS.

Setting up your Bugzilla Component

FIXME I think this information is probably completely wrong or Ie misunderstood it completely it was a long time ago . . . DB

You need a Bugzilla component so that users of your language can report errors, follows these instructions to create one for your language.

This information courtesy of Christian Rose. You should return these details to him at: menthos at gnome oeg.

You need to supply:

- language code
- language name (in English)
- language name (spelled in the language itself. We actually don't use this info in Bugzilla but on the http://www.gnome.org/i18n/ page. Please replace non-ASCII characters with proper HTML escape sequences. See the HTML source code of that page for examples)
- default owner (must be a valid bugzilla account). The default owner is the person who should be assigned the bugs by default. If he or she doesn't have a bugzilla account, he or she can create one at http://bugzilla.gnome.org/createaccount.cgi.
- default qa contact (must be a valid bugzilla account). The default QA contact is usually the person who should make sure the bug was fixed properly by the assignee. If the qa contact person doesn't yet have a bugzilla account, he or she can create one at http://bugzilla.gnome.org/createaccount.cgi. This field is optional, you don't need to decide on a default qa contact if you don't want to.
- component description. Usually of the form "Here you can place your bugs about $LANGUAGENAME [$LANGUAGECODE] translations". Example: "Here you can place your bugs about Swedish [sv] translations". If you have the possibility, try also to translate this into ASCII-only English, and we'll use the translation as well.

You have the option of assigning this the bug reports to a mailing list:

If you want, there's also the possibility to use a mailing list instead of an individual for the default owner and/or default qa contact fields. It's a bit more complicated; among other things you need access to the mailing list configuration. Here is what you should do if you want a mailing list in one or both of the fields above:

- Create a bugzilla account for your mailing list, i.e. a Bugzilla account with your list's address as account name.
- Subscribe the bugzilla daemon address (bugzilla-daemon@widget.gnome.org) to your mailing list, but also disable *ALL* mail from the mailing list to this address (If it's a Mailman mailing list you can change bugzilla-daemon@widget.gnome.org's mailing list options to NOMAIL).

Application Specific

There are some applications that need specific treatment. These are those:

gdm2

The login manager needs patches to gui/gdmlanguages.c and config/locale.alias to add your languages. Email your patch to "George"

Suggested bug report and related email for adding English (Canadian), use as a reference:

- http://mail.gnome.org/archives/gnome-i18n/2004-February/msg00256.html
- http://bugzilla.gnome.org/show_bug.cgi?id=135053

Also Arabic issue highlights how it all fits together:

- http://mail.gnome.org/archives/gnome-i18n/2004-March/msg00177.html

Actual CVS diffs to add Afrikaans, Northern Sotho and South African English

- http://cvs.gnome.org/viewcvs/gdm2/config/locale.alias?r1=1.38&r2=1.39
- http://cvs.gnome.org/viewcvs/gdm2/gui/gdmlanguages.c?r1=1.41&r2=1.42

Translating Documentation

Update

A large number of GNOME docs are now available for translation, via the gnome-doc-utils package, in both XML and PO format. This number is increasing steadily. We can look forward to having all GNOME docs available in both formats. Here is the current list:

http://kvota.net/doc-l10n/by-modules.html

The modules are listed alphabetically. You can see the POT (template file, all original strings but no translations yet) at the top of each module listing. Then the current translations are listed. Thus you can start with the POT, if there isn't a translation for your language yet, or update the current file. (Making sure you co-ordinate this with the translation team for your language, so effort is not duplicated.) As the original documentation is updated, so is the POT, and so are the existing translations. Just like the application PO files listed under your language on the GNOME l10n status pages.

Older information

On the gnome-i18n mailing list Christian Rose says, "At the moment, we don't translate documentation the same way we translate the user interfaces (i.e. with "po" files). However, we hope to do so at some point, since po files provide several essential advantages compared to maintaining translations of plain XML. One such advantage is that it divides documents into smaller pieces (messages), allowing you to see exactly what parts have an inconsistent translation and need updating."

"For the moment, what you may want to do is to use the "xml2po" utility in the "gnome-doc-utils" package/module. This will allow you to transform the XML/DocBook source of a document into a pot file that you can translate and maintain. Also, it allows you to reverse the process and create a translated XML file out of the po file later on."

Finding PO files

Technical notes on finding PO files via http://l10n.gnome.org/ from Simos Xenitellis on the translate-pootle list:

You can get .po files from the statistics pages. Have a look at http://l10n.gnome.org/gnome-2.10/index.html Click on individual language pages and they will lead you to the .po files. These files are updated daily, so the "resolution" of freshness is just one day (not bad).

FIXME The following script won't work anymore. We use the following scripts while making a translation memory IN-A-GLANCE:

```
% wget -O desktop.html http://l10n-status.gnome.org/gnome-2.10/el/desktop/index.html
% wget -O developer-libs.html http://l10n-status.gnome.org/gnome-2.10/el/developer-
↪libs/index.html
% grep 'el\.po' desktop.html developer-libs.html | awk -F\" '{print $6}' | sort |␣
↪uniq |awk -F\/ '{print $4}' | awk '{printf "wget -O GNOME210-%s http://l10n-status.
↪gnome.org/gnome-2.10/PO/%s\n", $1, $1}' | sh
```

David Fraser has also created a script that pulls the files out of CVS after finding them on the l10n-status web page

```
#!/bin/bash
  lang=$1
  branch=2.10
  export CVSROOT=:pserver:anonymous@anoncvs.gnome.org:/cvs/gnome
  if [[ $lang == "" ]]
   then
    echo syntax $0 lang
    exit
   fi
  [[ -d $lang ]] || mkdir $lang
  cd $lang
  [[:f_desktop.html]] && rm -f desktop.html
  [[:f_developer-libs.html]] && rm -f developer-libs.html
  wget -O desktop.html http://l10n-status.gnome.org/gnome-$branch/$lang/desktop/index.
↪html
  wget -O developer-libs.html http://l10n-status.gnome.org/gnome-$branch/$lang/
↪developer-libs/index.html
  pofiles=`grep ${lang}'\.po' desktop.html developer-libs.html | awk -F\" '{print $6}
↪' | sort | uniq | awk -F\/ '{print $4}'`
  for pofile in ${pofiles}
   do
    basefile=`basename $pofile .${lang}.po`
    actualbranch=""
    for possiblebranch in HEAD gnome-${branch}
     do
      branchext=`echo $possiblebranch | sed 's/[.]/-/g'`
      isbranch=0
      echo $basefile | grep $branchext >/dev/null && isbranch=1
      if [[ $isbranch == 1 ]]
       then
        basefile=`basename $basefile .$branchext`
        actualbranch=$branchext
       fi
     done
    # this would get it straight off the web page:
    # wget -O ${pofile} http://l10n-status.gnome.org/gnome-$branch/PO/${pofile}
    # this checks it out of CVS:
    if [[ $actualbranch == "" ]]
     then
      cvs -z3 co $basefile/po/$lang.po
     else
      cvs -z3 co -r $actualbranch $basefile/po/$lang.po

    fi
   done
```

1.10.3 KDE

The KDE Linux desktop is easily localisable and has a very mature localisation project.

Resources

- Localisation Project website

Suggested Priorities

These are the priorities as suggested by the SkuleLinux project.

There is no priority within each of these groups

FIXME this is probably out of date since the move to SVN version control

First	Second	Third	Fourth
kdebase	kdeaccessibility	kdeaddons	kdeextragear
kdelibs	kdeadmin	kdeartwork	kdeextragear-1
kdepim	kdeedu	kdegames	kdeextragear-2
koffice	kdegraphics kdekiosk kdemultimedia kdenetwork kdesdk kdeutils	kdetoys docs	kdenonbeta others

Of course you will probably want to move some applications found in the kdeextragear modules up. Or translate them when requested. Suggested candidates for bumping up include: amaroK, kbabel, superkaramba, Scribus, Kopete, Quanta+, Kaffeine, Juk, KDevelop, digiKam, kdebluetooth, kontact and kimdaba

Compiling your language

FIXME update for SVN

If you have made changes or wish to install the latest translations for your language from CVS do the following after checking out the kde-i18n module from CVS.

```
$ cd kde-i18n
$ echo XX > inst-apps # Where XX is your language's ISO code
$ make -f Makefile.cvs
$ ./configure # or ./configure --prefix=/usr if you do not want files installed is /
↪usr/local
$ make
$ su -c "make install"
```

Updating all translations manually

You can update all translations manually instead of waiting for KDE's automatic update bot called scripty.

```
$ cd kde-i18n
$ cvs up templates
$ VERBOSE=yes make -f Makefile.am.in merge
```

If you specify your language in a file called "inst-apps" then it will probably only update your language, which is a good thing. Have not checked this though.

Committing to CVS

FIXME fix for SVN

Always check the PO files by running:

```
$ cd kde-i18n
$ check_po_files xx
```

Where xx is your language code. Also running msgfmt in check mode

```
$ msgfmt --check -o /dev/null some-file.po
```

Checking accelerators

You now need to add the following to your kdeglobals in order to use the `F12` accelerator check feature:

```
[Development]
CheckAccelerators=F12
```

1.10.4 XFCE Translation

XFCE is a lightweight desktop environment and is therefore much faster then the more feature rich *KDE* and *Gnome* desktops.

Resources

- Translation Mailing list and archives
- Translation statistics
- Translation instructions

What first?

XFCE4 contains approximately 3500 words so targeting is not that much of an issue with XFCE itself. However, XFCE makes use of GTK therefore to appear fully translated you will have to translate some of those packages and a narrowed focus in that area would be useful.

FIXME: create a targeted list based on standard XFCE with no external applications.

1.10.5 Mozilla Product Localisation

Mozilla is the organisation credited with giving the world the Firefox web browser, Thunderbird e-mail client and many more. Most of the Mozilla products can be localised and this page (and pages linked to) aims to be a comprehensive resource of information about this process.

This page gives a broad overview of the localisation processes involved when localising Mozilla products using PO files, the Translate Toolkit and other PO based tool such as Pootle and Virtaal.

Product-specific information are on the projects' own pages:

- *Firefox*
- *Thunderbird*

Localising Mozilla on Pootle

FIXME work in progress, we'll cleanup and retire the other information and the Pootle way will be the only one that we talk about

So you want to translate Firefox into your language. To start is simple, to complete is a bit harder.

Before you begin lets be realistic about how much effort is required. If you can translate 1,500 words a day, like most professional translators, you can expect to complete Firefox in 20 days if you work fulltime. But you will still need to review the translations and perform in product review. So this is a lot of work.

We're helping to reduce that effort. While you translate we'll build the files needed by Firefox for you as we do for our South African and African languages. This takes a lot off your shoulders and you can happily begin translating without any knowledge of what happens at Mozilla, the files formats, etc.

What do I need to do to get started

1. Check that nobody else is doing your language. If they are, please join them. If not we're happy to host your translations.
2. Create an account on Pootle.
3. Ask us to add your language. We'll enable your language, give you rights to enable others and leave you to translate.
4. Begin translating the user1 component (the others look shorter and easier but please don't waste your time)

What should I translate next?

We have developed a number of modules that allow you to focus on the most important strings first.

How do I get my translations into Firefox

1. First, complete the key translations needed for Firefox
2. Apply for your language to be included
3. You will need to respond to a number of bugs that need fiing for your language. We can help you here, for most minority languages taking American English defaults is the correct approach.

Translation file formats used by Mozilla

Mozilla products use monolingual translation formats and is further discussed *here*.

Localisation work-flow

The Mozilla L10n project page contains the rest of the information needed to start a new localisation or join an existing localisation team.

- **Preparing translation files:** POT files for any string frozen version of Firefox or Thunderbird can be downloaded from the Mozilla l10n server or you can create it yourself. You can then create updated PO files with pomigrate2. These scripts can help to keep a repository of Mozilla PO files up to date and ready for translation.
- **Translating the PO files:** The PO files can now be translated using your favourite CAT tool, like Pootle for sharing the workload in a team or Virtaal to translate off-line.
- **Check for product specific issues:** The *Firefox* and *Thunderbird* pages describe some of these issues.
- **QA and bug checking:** You need to test your build. Probably the best preventative step you can take it to run pofilter's *variable*, *accelerator* and *escapes* tests. Errors that these tests pick up are all capable of introducing potential breakage that is hard to trace.

- **Preparing translation files for uploading:** The PO files now need to be converted back to the original monolingual files before they can be committed to the Mozilla repositories. The po2moz tool can be used to recreate the original file structure using the new translations.

Useful odds and ends

Dialogue Sizes

Various dialogues are sized within the XUL work. Often they have entries in the DTD which allow you to change the size of the dialogue. It is a very time-consuming process, but hopefully the bookmarklet and instructions created by Axel Hecht will help you quickly get the correct size to enter into your final files.

For reference:

- Put the following text into the location field of a new bookmark (therefore won't work with Thunderbird): (Make sure that you are not putting in smart (round) quotes produced by the wiki software)

```
javascript:(function(){var p=2;var cs=window.getComputedStyle(document.
↪documentElement,null);var fs=cs.getPropertyCSSValue('font-size').
↪getFloatValue(5);var w=cs.getPropertyCSSValue('width').getFloatValue(5);var␣
↪h=cs.getPropertyCSSValue('height').getFloatValue(5);prompt('Dialog size:',
↪'width: '+Math.ceil(w/fs*p)/p+'; height: '+Math.ceil(h/fs*p)/p+';');})();
```

- Browse to the correct XUL file with Firefox, for example `chrome://browser/components/preferences/preferences.xul`.
- Choose your bookmark with the JavaScript and a message box displays the width and height.

Requirements for official build

- See https://wiki.mozilla.org/L10n:Becoming_an_Official_Localization
- Translate the start snippets

Locale Switcher

This multi locale switcher is best for use in cases where you will have more than one language to install. Even with two it is much cleaner then other locale switchers available.

Releasing

There are some requirements which stem from the trademark policy. Mozilla products can be released as either an official build or a community release.

Links

- Mozilla L10n Home
- Localisation policy/trademarks, etc (2)
- Scripts used by Translate.org.za

1.10.6 Firefox

Firefox is the web browser that was born out of the *Mozilla Product Localisation* project. It is now their default target and is renowned for its security.

This page contains Firefox-specific localisation notes and issues. You should read this in conjunction with the *Mozilla Product Localisation* pages as there are some things shared between Firefox and *Thunderbird* localisation.

As stated on the *Mozilla Product Localisation* page, there are more complete instructions about the requirements and processes involved in Mozilla localisation on the Mozilla L10n project page.

Links

- Firefox localisation home page on Mozilla wiki
- Building a Native Installer under Windows XP
- Requirements for official build: Firefox Extras

Settings

These are configurations not customisations. These are things that you **must** change. Remember that you need to go through a formal procedure of review to commit changes to some of these files in the Mozilla repository, especially a file like browser/chrome/browser-region/region.properties. This usually means you need to file a bug, ask for review by setting flags and more things and put people in CC. Ask at Mozilla for the newest rules.

1. file: `toolkit/defines.inc.po`
   - entry: `MOZ_LANG_TITLE`
   - use: Used in the language .xpi in file `install.rdf`. Its added to *%s Language Pack*
   - action: Set to the English name for your language
2. file: `browser/defines.inc.po`
   - entry: `MOZ_LANGPACK_CREATOR`
   - use: Used in `install.rdf`'s creator tag
   - action: Set to the name of the language pack translator eg. a group, company or person
3. file: `browser.inc`
   - entry: `MOZ_LANGPACK_CONTRIBUTORS`
   - use: Credit those who helped create the language pack
   - action: you need to edit this file directly ie. not the PO file. Uncomment the `MOZ_LANGPACK_CONTRIBUTORS` line and make changes as needed.
4. file: `toolkit/chrome/global/intl.properties.po`
   - entry: `general.useragent.locale`
   - use: FIXME Assume this is the locale the browser publishes.
   - action: change to your locale `xx-YY` (language-COUNTRY) leave out country if you don't need that for your language
5. file: `toolkit/chrome/global/intl.properties.po`
   - entry: `intl.accept_languages`

   - use: tells a website what languages you prefer your content in
   - action: set the various language settings you require

6. file: `browser/chrome/browser-region/region.properties.po`
   - entry: `general.useragent.contentlocale`
   - use: unsure
   - action: change it to your country code

Customisation

You can change various settings to make the browser more specific to your locale. Be aware that some changes are not allowed in official Mozilla Foundation builds.

TODO

- Default start URL
- Default search engine
- Adding search engines specific to your locale
- Bookmark customisation
- Browser identification
- Accepted languages

Other things to localise

- Web parts. See http://wiki.mozilla.org/L10n:Web_parts
- Google snippets. See this bugzilla bug

This extract of an e-mail from Pavel Franc of the Czech team highlights other pieces of Firefox that need to be localised:

```
The Czech (cs-CZ) Thunderbird 1.0 windows builds are ready.
We do now our QA testing.

ftp://ftp.czilla.cz/test/thunderbird/1.0/

The following files outside chrome were changed in zip archive:

  * * README.txt ... translated
  * * talkback-l10n.ini ... translated
  * + cs.aff ... Czech spellchecker
  * + cs.dic ... Czech spellchecker
  * - en-US.aff ... removed
  * - en-US.dic ... removed
  * * rss.rdf ... translated
  * * mailViews.dat ... translated
  * * all-thuderbird.js ... changed useragent.locale

and in installer:
  * * install.js ... changed locale and translated shortcuts
  * * UninstallThunderbird.zip ... translated
  * * 7zSD.sfx ... translated by ResourceHacker
```

1.10.7 Thunderbird

Thunderbird is the e-mail client that was born out of the *Mozilla Product Localisation* project.

This page contains Thunderbird-specific localisation notes and issues. You should read this in conjunction with the *Mozilla Product Localisation* pages as there are some things shared between Thunderbird and *Firefox* localisation.

As stated on the *Mozilla Product Localisation* page, there are more complete instructions about the requirements and processes involved in Mozilla localisation on the Mozilla L10n project page.

Links

- Packaging a localised Max OS X version.

1.10.8 OpenOffice.org

What to do first?

OpenOffice.org is a large project so it is helpful to know what sections to translate first so that your effort is optimised. The following modules are listed in order based on the assumption that you want to start by translating OpenOffice.org Writer. We have ignored the Spreadsheet, etc and the installer.

This is based on version 1.1.3 and will be slightly different for 2.0

The syntax is that used by podebug which is used to create a tagged version of OpenOffice.org. The tags a listed below and follow this convention:

- the first word is the base directory
- the second is the first letter from each subsequent directory and
- the last word contains the first letters of the actual PO file.

Installer

Setup Wizard

setup-su-page
setup-s-uiba
vcl-s-src
read-d-read
setup-sc-jvms

Setup Wallpaper

setup-su-dial
setup-s-ui
setup-s-uiba
setu-su-page

Registration

svto-s-dial
vcl-s-src

Office Shell

svx-inc
offm-so-app
scp-s-calc/impr/draw – doc types under new menu

Writer

General

svx-inc
sw-su-inc
sw-su-utlu
sw-su-app
sw-sdi
svx-sdi

sfx2-sdi
sfx2-s-doc
sfx2-s-appl
offm-so-intr
sfx2-s-dial
offm-sdi
svx-s-dial
sfx2-s-menu
sw-su-shel
svx-s-stbc

Format Char/Paragraph/Page

sw-su-chrd
svto-s-cont
svx-s-dial
svx-s-tbxc
sfx2-s-appl
vcl-s-src
sw-su-app
sw-su-utlu
sw-su-fmtu

sfx2-s-dial
sw-su-misc
sw-su-frmd

Footnotes/Outline numbering

sw-su-misc
sw-su-utlu

Paste Special

so3-src
vcl-s-src

Open/Save file dialogue, Export to PDF

svto-s-file
svto-s-cont
vcl-s-src
sfx2-s-appl
svto-s-misc
offi-rdoo-Offi
sfx2-s-doc
sfx2-s-dial

Printer

svto-s-dial
vcl-s-src
padm-sour (admin config)

Options

offm-so-dial
svx-s-opti
setu-su-page
sfx2-s-dial
sfx2-s-appl
svx-s-dial

Text Document

sw-su-conf

svx-s-opti

Help (only the help browser not the content)

sfx2-s-appl

Others FIXME

OpenOffice.org is the leading cross-platform Office suite. Its a large project and a large localisation undertaking, but it is an important component of a localised desktop.

Resource

- KhmerOS – These pages prepared by Javier SOLA are by far your best resource for localising OpenOffice.org
  - Localisation Guide to OpenOffice.org 2.0
  - Making OpenOffice.org 2.0 locale files
  - Creating your languages collation sequence for OpenOffice.org 2.0
  - Some localisation tips for OpenOffice.org 2.0
- Current and in progress localisations

What is your language's LCID

Microsoft defines LCIDs for various locales. You need to know this so that OpenOffice.org can work well on Windows and also so that documents you create can move seamlessly between MS Word and Office Writer as the language identifier is correct.

There are a number of places that you can use to identify the LCID. For most languages they will all agree but in some cases (See 1072/Sutu/Sesotho) it helps to look at all list to help clarify what exactly Microsoft meant.

- List of Locale ID (LCID) Values as Assigned by Microsoft
- Microsoft Word 2000
- Windows XP/Server 2003 includes XP service pack 2
- National Language Support Constants much clearer layout and mentions codepage. Doesn't seem completely up to date though.

What to do first

This is a very large application. If you can do a smaller section of the total and still have a useful product then that will help. We created this *rough targeting guide* using OpenOffice.org 1.1.3 and podebug

Localisation

Read the localisation documentation on the OpenOffice.org website: http://wiki.services.openoffice.org/wiki/Category:Localization

Things are now very easy since they are using Pootle. You can translate online on Pootle, or download the files to work offline with something like Virtaal.

gsicheck

The OpenOffice.org guys have a tool for checking the SDF file called gsicheck. But of course you don't want to build the whole of OpenOffice.org simply to get one tool. pofilter will pick up most errors that gsicheck does but its nice to know that your SDF is good before submitting it. Read more and download from the OpenOffice.org website:

http://wiki.services.openoffice.org/wiki/Gsicheck

Then install it and use it

```
tar xvzf gsicheck-1.7.8_2.0m122.tar.gz
cd gsicheck-1.7.8_2.0m122
./gsicheck -c <GSI/SDF file>
```

Now go and fix the errors that it detected. You should correct these in your PO files.

AutoCorrect

The OpenOffice.org AutoCorrect file is a zip file called for example, acor_en-US.dat. Søren Thing Pedersen has created csv2acor.py which generates an AutoCorrect file from CSV sources.

The autocorrect file contains 3 XML files:

- DocumentList.xml – pairs of mistyped words and their correct spelling
- SentenceExceptList.xml – abbreviations that end with a fullstop that should be ignored when determining the end of a sentence
- WordExceptList.xml – Words that may contain more than 2 leading capital eg. CDs

When using csv2acor.py your need to have 3 files with the same name as above but with a .csv file extension. WordExceptList.csv and SentenceExceptList.csv contain just a list of entries one per line surrounded by double quotes ("). DocumentList.csv is a comma separated list with the mistyped word in the first column and the correct word in the second column, all also surrounded by double quotes.

The translation program Virtaal also makes use of these files, so consider contributing it to this project as well.

WordExceptList.xml

If you have an existing spell checking wordlist then use the following to extract potential words:

```
egrep "^[A-Z][A-Z][a-z]" spell-wordlist > WordExceptList.new
```

This extracts all words that start with two capitals followed by a lower case letter. Add all the characters valid in your language.

SentenceExceptList.xml

If you have an existing spell checking wordlist then use the following to extract potential words:

```
egrep "\.$" spell-wordlist > SentenceExceptList.new
```

This extracts all entries that end in a fullstop.

DocumentList.xml

If you have an existing DocumentList.xml you can convert it to CSV using the following:

```
sed "s/<block-list:block block-list:abbreviated-name=\"/\"\\n\"/g;s/\" block-
↪list:name=\"/\",\"/g;s/\"\/>//g" < DocumentList.xml > DocumentList.csv
```

Your'll need to edit DocumentList.csv to remove some of the remaining XML data.

A cleaner method is to use the following XSLT – this way you don't have to clean any XML data (so this is suitable for batch mode):

```
<?xml version="1.0" ?>

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 version="1.0"
 xmlns:block-list="http://openoffice.org/2001/block-list">

 <xsl:output method="text" encoding="utf-8"/>

<xsl:template match="//block-list:block">
  <xsl:text>"</xsl:text>
  <xsl:value-of select="@block-list:abbreviated-name"/>
  <xsl:text>"</xsl:text>
   <xsl:text>,</xsl:text>
   <xsl:text>"</xsl:text>
   <xsl:value-of select="@block-list:name"/>
   <xsl:text>"</xsl:text>
   <xsl:text>&#x0a;</xsl:text>
  </xsl:template>

</xsl:stylesheet>
</xml>
```

Run this script through any XSLT processor, e.g., for Saxon, type:

```
java -jar saxon8.jar DocumentList.xml <name-of-xslt> >DocumentList.new
```

Generating your new AutoCorrect file

Then run csv2acor.py acor_xx-YY.dat where xx-YY is your language and country code.

Spell Checker and Hyphenation in the official build

In order to add your spell checker and hyphenation file to OpenOffice.org CVS you need to do the following:

- Ensure your license is compatible
- Fill in the form at http://external.openoffice.org/
- Fill out an Issue assigned to mh who needs to process the approval for inclusion

Holidays

- wizards/source/schedule/LocalHolidays.xba

Looks like a StarBasic program that allows you to specify holidays, etc. FIXME need to check this more carefully

Child Workspace

OpenOffice developers use what they call child workspaces to make fixes and commit changes. These are usually linked to related bugs in IssueZilla.

Here some instructions to help you track your changes and see if they have been integrated/fixed:

- go to: http://eis.services.openoffice.org/
- log on with your openoffice account. Example coni@openoffice.org, and password
- click Childworkspaces
- click Search
- enter localisation% in the Name field
- wait. . . .

Now you see which l10n CWS have been integrated and which not. By clicking on the CWS name you see the list of the bugs registered to that CWS. Once approved by QA you'll exactly know in which milestone the CWS has been integrated.

1.10.9 Debian Translation

Debian is a Linux distribution.

Resources

- http://cvs.alioth.debian.org/cgi-bin/cvsweb.cgi/debian-installer/doc/translations.txt?rev=1.24&content-type=text/x-cvsweb-markup&cvsroot=d-i
- Main Installer progress page:
- Example progress for Dutch
- 2nd stage translations
- An old document outlining the Debian process
- Debian localisation stats

1.10.10 Translating Fedora

Fedora Core is the community version of Red Hat Linux. They are both popular Linux distributions.

Resources

- Mailing list
- Bugzilla
- CVS account request form
- Translation stats (2)
- Translation FAQ

What to Translate First

Fedora uses the PO format for translation which makes it quite standard. For your language to be included you need to reach 90% on the essential files. The essential file list includes these packages:

- anaconda/anaconda-online-help/gui (help-screens-C)
- anaconda/anaconda-po (anaconda)
- comps-po
- firstboot
- rhgb
- redhat-config-packages (It seems that this has been removed from the essential file list)
- redhat-artwork
- redhat-menus

Also look at the *Direction setting based on the installation and bootup process* process to see other packages that might be important.

Release Schedule

Fedora follows a six monthly release schedule which can be viewed here.

Red Hat releases are more conservative and will draw on translations found in the Fedora Core files.

Starting

As usual check on the mailing list to see if anyone is translating your language. If someone is then work out a way to combine forces. If not you will probably become the translation coordinator for your language. Either way you might want *CVS access*. In order to translate you don't need CVS access. But you will as soon as you want the data in the applications.

Getting CVS access

Browse to http://i18n.redhat.com/cgi-bin/i18n-signup where you will have to supply your ssh public key. Use:

```
$ ssh-keygen -t dsa
```

if you don't have one. This key is found in

```
$HOME/.ssh/id_dsa.pub
```

just copy and paste it into the form.

Getting the files via CVS

To check out the translate module

```
$ cvs -d :ext:YOURUSERNAME@elvis.redhat.com:/usr/local/CVS  co translate
```

Note: the CVS directory is structured in the GNU way. Ie you have a POT file and all of the $LANG.po files in one directory. You may want to make a few scripts to allow PO and TEMPLATE directories per language if that is the way you are used to.

All files are in HEAD except system-config-packages which uses the redhat-config-packages-1_1_x branch for Fedora Core 3.

Per Language Checkout and Update scripts

Tools use to update POT and PO files for Fedora on a per language basis which is nicer as otherwise everything is thrown at you.

- http://mystery.lviv.net/~lvm/fedora-cvs/fedora-cvs-0.3.tar.gz

The tools look like they are from the early days of the Fedora so they might need updating to be relevant to the latest layout in CVS.

Enabling installation in your language

Once you've completed about 90% of the essential modules you can ask for your language to be included in the installation process.

File an enhancement request to get your language accepted as an anaconda install language.

Here is an example: https://bugzilla.redhat.com/bugzilla/show_bug.cgi?id=118028

You need to make changes to `lang-table` and `lang-name`

Checking

Red Hat treats translations as part of a component not as a global resource. So you will probably have to check that each item you have translated is included in the final RPM. You can either install the application and validate or download the beta SRPMS and check. Installing is easier but requires a large amount of bandwidth. Downloading the SRPM requires some RPM building experience or at least the ability to read an RPM spec file to check the contents or the ability to check the tarball within the SRPM to validate that your languages are included.

Additional

You might want to check on the following items which relate to localisation:

- redhat-config-languages - check that your language is listed in locale-list. - see http://bugzilla.redhat.com/bugzilla/show_bug.cgi?id=107450

1.10.11 Mandriva

Mandriva is a Linux distribution.

1.10.12 Kickstart

- Subscribe to cooker-i18n mailing list, as per the instruction from Mandriva localization page.
- If there is existing language team, send translated files to team coordinator.

- If there is no existing language team, send to mailing list. After certain amount of contribution, you will be awarded SVN access.

1.10.13 Resources

- Main localisation page
- Cooker-i18n mailing subscription
- Mandriva's SVN guide for applications translators
- Mandriva's documentation project also welcomes translators
- Unofficial localisations statistics

1.10.14 SUSE

The openSUSE project is a worldwide community program sponsored by Novell that promotes the use of Linux everywhere. The program provides free and easy access to openSUSE. Here you can find and join a community of users and developers, who all have the same goal in mind — to create and distribute the world's most usable Linux. openSUSE also provides the base for Novell's award-winning SUSE Linux Enterprise products.

Resources

- openSUSE Localization portal Where you will find resources for practical translation work
- Mailing list (Subscribe to m17n (multilingualization))
- m17n List Archives
- Localised files FTP server

1.10.15 Wikis

MediaWiki

MediaWiki is the wiki used by Wikipedia to host their collaborative encyclopedias. So this is no ordinary localization of a wiki.

Resources

- Localization Stats
- MediaWiki localisation

This is a summary of localisation information for various wiki software

Wiki	GUI	Content
DokuWiki		Plugin
MediaWiki		

1.10.16 Friendly howto for translators

XXX

XXX

Working with.po files and The Translation Project

I've written this to try and meet a need I certainly had, as a translator starting with the TP. :-)

Sections marked **!** are Translate Tips!

Our translation toolbox

A true craftsperson respects his or her tools. We are very fortunate in having some excellent localization tools nowadays. In order to translate a .po file, you need:

The latest version of gettext

Check this with:

```
``gettext --version``
```

Currently, the answer is:

```
``gettext (GNU gettext-runtime) 0.16``
```

but since this entry will age, it's worth checking the latest version anyway. Whether you have an older version, or no version at all, you will need to install the current version of gettext. Fink doesn't always list the latest version, so please check:

http://ftp.gnu.org/gnu/gettext/

and download and install the latest version there.

? *gettext manual:* completing gettext installation

A translation tool

You can simplify your task considerably by using a dedicated .po editor. These are currently:

- Virtaal (Windows+Linux)
- kbabel (KDE)
- gtranslator (GNOME)
- emacs (various platforms, including Mac OSX GUI)
- poedit (various)

People

People have put a lot of effort into helping us with our task. We also have available:

- The Translate Toolkit, which contains a number of really useful tools
- Translate Documentation, including this doc
- The web-based .po editor Pootle: excellent for sharing the load
- po4a, a Debian effort which includes many conversion tools

The first three are available from Translate at Github.

! With po4a and the Translate Toolkit, you can turn practically anything into or out of the .po format, possibly including organic forms… :-)

? also see other sections of this wiki, for key resources and helpful information on many aspects of translating. Experienced translators have spent a lot of time contributing to this wiki, to save all of us time and hassles. Please feel free to add your experiences at any time: this is our shared resource. :-)

Let's have a look at the translation task…

So you want to translate a .po file

A PO file (Portable Object) is a format created especially for what we need to do: grab the info, and move it about. You can put it through any text editor, use just about any encoding (although utf-8 is safest for other reasons), feed it to your dog, but it will come out the other end as exactly what it was to start with … a text file.

! The only thing that differentiates a .po file from a normal .txt file is the headers, and the structure of each string block. This is checked by the TP robot program when you submit the file, so knowing what to get right can save you re-submitting it (and being talked down to by a robot, which can be a bit exasperating. :-)

Headers

Here is a blank set of .po file headers:

```
# SOME DESCRIPTIVE TITLE.
# Copyright (C) YEAR THE PACKAGE'S COPYRIGHT HOLDER
# This file is distributed under the same license as the PACKAGE package.
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
#
#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"POT-Creation-Date: 2003-07-24 09:35+0200\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: 8bit\n"
```

You will usually find a complete blank set, like this, in a file which has not yet been translated at all, a .PO Template file, with the file extension .pot.

! All you need to do to change a .pot to a .po is fill in the headers and change the name of the file to .po. It's as easy as that.

You may find some of the headers still blank, or not up-to-date, when you are updating a partially-translated, or out-of-date file.

! So you always need to check the headers: do this first, do it last (before submitting the completed file), and you'll save yourself hassle.

There are two headers which may or may not appear in that block, but it's better if they *do* appear. You can add them yourself:

```
"Report-Msgid-Bugs-To: \n"
```

and

```
"Plural-Forms: nplurals=INTEGER; plural=INTEGER\n"
```

so here we have a complete set (note the positions of those two additional headers):

```
# SOME DESCRIPTIVE TITLE.
# Copyright (C) YEAR THE PACKAGE'S COPYRIGHT HOLDER
# This file is distributed under the same license as the PACKAGE package.
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
#
#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2003-07-24 09:35+0200\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: 8bit\n"
"Plural-Forms: nplurals=INTEGER; plural=INTEGER\n"
```

Each header has a job to do, so let's go through them one-by-one:

The title header

```
# SOME DESCRIPTIVE TITLE.
```

is there to give quick information as to the title of this package. Here you input the *name* of the program (not the version number). I'll use the program Tuxpaint (an excellent art program for young children), and my language, Vietnamese, as the example in this section.

```
# Vietnamese translation of TuxPaint.
```

! Note that all these headers have *a # sign and one space* before the information. The robot is very picky about this, as it is gettext's way of signifying an informative header. gettext actually parses this information, and the whole file, so by getting the format right, we save ourself time spent fixing the errors, when the file won't parse.

The copyright header

```
# Copyright (C) YEAR THE PACKAGE'S COPYRIGHT HOLDER
```

In the case of packages sent to the Translation Project, the software is usually open-source, free software, so the information here is usually (I'll use this year):

```
# Copyright © 2005 Free Software Foundation, Inc.
```

If you can access the copyright sign © fairly easily from a keyboard layout or special characters' input feature, it does look more professional. ;-) (It's typed Right `Alt+C` on a qwerty international keyboard.)

Occasionally, a file will come with a proprietary copyright header: somebody has created, and claims copyright over this file (for example):

```
# Copyright © 2001-2005 Nguyn Th Hoa.
```

In this case, you respect the header already there. Do not change it.

! If your file has a proprietary copyright header, and is rejected by the TP robot for not having a FSF copyright header, simply write to the TP co-ordinator at:

translation@iro.umontreal.ca

because that is their problem, not yours, although it's rather annoying to get your file rejected for something that isn't under your control. The co-ordinator needs to set an option for these files so they won't be rejected next time you, or another translator submits them. Again, by contributing what we can at the time, we all help each other. ^_^

Associative copyright header

```
# This file is distributed under the same license as the PACKAGE package.
```

This header (not always present, although it should be) releases the translation under the same copyright as the original file. This saves queries about the copyright of translations, and if you are volunteering for the TP (Translation Project), you will have already filled out a disclaimer which assigns your copyright to the FSF. This saves a lot of hassle, simplifying the copyright issues for everybody.

All you need to do here is insert the package name again:

```
# This file is distributed under the same license as the TuxPaint package.
```

The list of translators

```
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
```

This will only be blank if you are the first person to translate this file at all. If it has been translated, even partially, before, the names of any previous translators will each occupy one header exactly like this. So if there is only one translator (I'll use my name):

```
# Clytie Siddall <clytie@someserver.net.au>, 2005.
```

However, if there have been previous translators, there will be more than one translator header, for example:

```
# pclouds <pclowds@anotherserver.com>, 2002.
# Tran Minh Thanh <tmt@yahhooo.com>, 2004.
# Clytie Siddall <clytie@someserver.net.au>, 2005.
```

So in theory, you could have a lot of these headers, one after the other, but in practice, there are one to five translator headers.

! Don't change any of the older translator headers, just insert your own below the newest one. These headers ensure that everybody who has put effort into translating this file, gets both some recognition, and must take responsibility, for their work.

The blank header

```
#
```

You may have a blank header line between the two sections of the file header. This makes it easier to read. You don't need to do anything here. ;-)

The fuzzy header

```
#, fuzzy
```

Note the comma after the # sign. This indicates that this header is read by gettext as *information* on the string blocks. If this header is present, there are incomplete or incorrect strings in this file. Your .po editor may remove it when you finish those strings, or, if you're using a text editor not designed to handle .po headers, you may remove it yourself. Just delete the whole line.

Fuzzies are strings which are incomplete or incorrect. gettext makes this judgement, for example, on whether the quotation marks, any variables and line-breaks match, or not. It will also base this judgement on whether any compendium (glossary) strings suggested by msgmerge match the original string completely, or not. Each *fuzzy* string is marked with the fuzzy header, and needs careful checking. More on that further down. 8-)

The gettext manual: fuzzy strings

The string pair

```
msgid ""
msgstr ""
```

This blank string pair indicates to gettext, I imagine, the structure of the strings in the file. The msgid string is the original text, and the msgstr is the translation.

! The output file must contain both, and they must be surrounded by quotation marks. Do not alter this header.

The package-version header

```
"Project-Id-Version: PACKAGE VERSION\n"
```

Here, the version of the package is important: it's a header you need to watch out for when updating a file.

! The TP robot requires the name of the program to be separated from the version by a space, not a hyphen or underscore. So this header may vary in that way, from the original file-name.

Original file-name: tuxpaint-2.1pre

```
"Project-Id-Version: Tuxpaint 2.1pre\n"
```

! Remember to change this header when you update a file.

Use all the information in the version part of the filename: 0.03a2, 2.01b, 0-03.2pre2, this is all useful information about the stage of development of this package.

- **a** means alpha, a very early release, usually quite unstable, for testing purposes only;
- **b** means beta, a later testing release, often quite stable, but not guaranteed or supported. You can learn a lot and help software development by testing beta software, especially for language support. :-)
- **pre** means pre-release, the last version(s) before a full release version: finished testing. It probably means the full version isn't far away, so you'll need to update the file again then.

If you're using the programs you translate, remember to check the version data to decide if the program is stable or needs further testing. If you decide to help test a program, that's great, as long as you don't expect it to be completely stable or have tech support. On the other hand, the developers and other people contributing, as you are, by testing, will be very happy to discuss the program and support each other on the program's mailing list. ;-)

The report-string-bugs header

```
"Report-Msgid-Bugs-To: \n"
```

This header is often omitted, or not filled-out, and this is a nuisance for us, because it's the contact address for us to use when an original string is incorrect (typo, missing bracket, missing words, bad grammar or spelling), or when we don't understand a string well enough to translate it.

It wastes our time if we need to go back to our team page, click on the file-name to go to its textual domain, then look for the homepage of the program or some other contact information; often you have to Google for quite some time, in order to find it at all.

When you find that contact address, please fill it in in your file, so the next person, quite possibly you :-) , won't need to waste time looking for it. It's a good idea to encourage your developers to fill in this header.

? One handy thing I've found out about these contact addresses is:

- all GNU packages have the contact address:

bug-PACKAGE_NAME@gnu.org

- all GNOME bugs are reported via Bugzilla
- all Debian bugs are reported via email to:

owner@bugs.debian.org

with the filename as the subject line, and the body starting with:

```
Package: FILENAME
Version: VERSION_NUMBER
Severity: wishlist
Tags: l10n, patch
```

The creation date of this file

```
"POT-Creation-Date: 2004-07-24 09:35+0200\n"
```

The .pot is the original, untranslated file, so that was when this version of it was created by gettext. Updated files will have .po creation dates.

This information is unimportant to you (you don't change it), except:

! you will have to make sure your revision date (the date of your changes to this file) is *after* the creation date, otherwise the TP robot will say "I object!" and you really can't blame it. We translators have not yet found out how to make time go backwards. LOL

The last-change date header

```
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
```

This is blank in an original .pot file, since no changes (translations) have occurred. In an updated file, a date will be present. All we need to remember, is:

! to update this date before submitting our completed file.

A .po editor program may do this automatically. You can do it manually at any stage. In BBEdit, you can create a glossary item using strftime variables (you can just save it and use it without having to understand how it works):

```
"PO-Revision-Date: #LOCALTIME %F %R%z#\n"
```

which, anytime you select that whole header, will replace it with your local time and UTC offset. In my case, that is, as I write this sentence:

```
"PO-Revision-Date: 2005-05-16 14:58+0930\n"
```

! Note the order of the date: year-month-day, the year being four numbers, the month two, and the day two. This means including leading zeros when the number is less than 10, as in the current month: 05 (May).

Note the UTC offset: +0930. This says that my timezone (Adelaide, Australia, Central Australian normal time, not daylight saving) is 9.5 hours, 9 hours and 30 minutes, after GMT or UTC time (00:00).

! You need to fill in your timezone here, and note that there is no space before it in this header. Remember the leading zero if, as in my case, you're less than ten hours before or after UTC. (BBEdit's glossary item, or your .po editor, may do all this for you.)

The most recent (last) translator header

```
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
```

Where you have been the *only* translator, your name will appear both in the First-Translator header, and here in the Last-Translator header, which may result in you feeling like the Only-Possible-Translator. LOL

All you need to do is fill in your name and address here, again, but don't include the year, as in the First-Translator header, because the PO-Revision-Date: header supplies that.

If a previous translator's name is filled in here, you need to edit that to show your name. Make sure that previous translator is mentioned in the top part of the headers (first, second, third, however many translators there have been).

So in my case, this header will show:

```
"Last-Translator: Clytie Siddall <clytie@someserver.net.au>\n"
```

The language-team header

```
"Language-Team: LANGUAGE <LL@li.org>\n"
```

Here is where your language team is given credit for all the hard work you do. It also supplies an alternative contact address for people writing to you about your translations. This is particularly useful when email addresses become outdated, as people move around or change their details.

Your language team will be the name of your language, and sometimes of the project. The address will often be the team mailing-list. So in my case, this header will be:

```
"Language-Team: Vietnamese <gnomevi-list@lists.thatserver.net>\n"
```

or

```
"Language-Team: Gnome-Vi <gnomevi-list@lists.thatserver.net>\n"
```

The MIME-version header

```
"MIME-Version: 1.0\n"
```

This will usually come filled-in. You don't need to worry about it. Isn't that great? :-D

The Content-Type header

```
"Content-Type: text/plain; charset=CHARSET\n"
```

! This is really important. It sets the character set for your language. UTF-8 is the best choice, but if your language requires another charset (character set), please input it here. I imagine this header will soon be filled in automatically as UTF-8. For my language:

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

God bless Unicode! It's such a relief to be able to shrug off all those clumsy, tortuous legacy encodings... Now we just need better Unicode support in all systems. 8-O

The Content-Transfer-Encoding header

```
"Content-Transfer-Encoding: 8bit\n"
```

This should also come already-set. If not, please input **8-bit**, which can handle UTF-8 and other complex charsets in transit. You don't want your hard work to be messed up in submitting the file, or when it is sent on to your developers.

The Plural-Forms header

```
"Plural-Forms: nplurals=INTEGER; plural=INTEGER\n"
```

This is often not included, but it *should be*. When you encounter plural (describing more than one person or thing) strings in your files, this plural header makes sure you have the correct number of fields to fill in with the translation. This varies considerably from one language to another. For my language:

```
"Plural-Forms: nplurals=1; plural=0\n"
```

because Vietnamese has no plural forms in that sense. One book, two book. But you should see our pronoun collection… 8-)

Some languages have several plural forms. A plural msgid looks like this:

```
msgid "Found and replaced %d occurrence."
msgid_plural "Found and replaced %d occurrences."
```

Since English, the original language, *does* have plural forms in this sense. If your language behaves like English in this way, you will have two msgstr fields to fill in, like this:

```
msgid "Found and replaced %d occurrence."
msgid_plural "Found and replaced %d occurrences."
msgstr[0] ""
msgstr[1] ""
```

but in my case, it should be:

```
msgid "Found and replaced %d occurrence."
msgid_plural "Found and replaced %d occurrences."
msgstr[0] ""
```

If your plurals header is set correctly, you will have the appropriate number and kind of msgstr fields to fill in. So it's a big help.

! Find out what yours is, and make sure you fill it in for all your files: it will save you hassle.

If you are unsure of the plurals header that should be set for your language, please consult your team leader – and if s/he is unsure, you can discuss this on the TP mailing list, an excellent place to ask questions and share experience.

And those are all the headers you need to complete! These headers all save time and trouble in the process of localizing an application. You can set them in your .po editor, or simply keep a copy of them to paste over the out-of-date or original headers.

! By getting them correct, and finding your own way to deal with them, you become a better translator, because the true craftsperson makes the best use of his or her tools. The .po format is one of our tools.

? *the gettext manual:*

the po format

filling in the header entry

Where do we get our files?

Your team page ((If your language does not have a team yet, please contact the TP co-ordinator about creating one.)) at the TP will list the files available to be translated. You need to ask your team leader which files need translating, or ask to translate particular files, and s/he will notify the TP co-ordinator that you are assigned to that file. Your name will appear next to it on your team page. What does becoming a TP translator involve?

To be a TP translator

You need to register with the TP. This is simple, although it involves one hold-up: the disclaimer.

- Your team-leader may email the TP co-ordinator

S/he will advise the co-ordinator that you want to join the project, or s/he may ask you to do it with his/her permission, but it is important that you are *part of the team*, so that's where the team leader comes in.

A language team can support each other, and ensure a consistent approach to the task. It's confusing, and much less effective, to have people working separately on the same language, not communicating or co-operating. The TP requires changes to go through your team co-ordinator, so there should be no conflicts or confusions over who does what, how and why. 8-)

! Check with your team-leader, who will be a big help to you, join the team mailing-list, and join the TP.

- Once you have registered with the TP

(yourself with your team-leader's permission, or through your team-leader), you need to fill out the TP disclaimer, sign it, and fax or post it to the FSF. (If you have any difficulty understanding the information, or with submitting the disclaimer, your team leader is there to help you.) You can also print the disclaimer form, sign it, scan it and email it. One way or another, this disclaimer needs to arrive at the Free Software Foundation, and be logged under your name. When this has occurred, your name on your team page will show:

	Disclaimer
Your name	Yes

The difference the disclaimer makes (apart from simplifying copyright issues as mentioned above, which is its reason for existence) is that most TP files are not available for translation unless your disclaimer is logged with the TP. When you go to a file's textual domain page (by clicking on its link on your team page), check down the page whether a disclaimer is required.

! Until your disclaimer is logged with the FSF, you can only translate non-disclaimer files, but there are quite a few of them, so don't hold back. ;-D

How do we get the most current files?

The files listed on your team page *should* be the most current files. Developers send them in to the TP to be translated, and they should be sent in automatically, each time they are updated. It is extremely important to translate the current file, otherwise, your translation may not be used at all, or won't be used by the majority of users. Downloading your file from your team page at the TP should ensure you get the latest, most current file.

! If it turns out that this file is not the most current (rare, but possible), please email the TP co-ordinator so this can be fixed.

Methods of establishing and maintaining currency include CVS, SVN and private repositories. The TP saves you the trouble of learning how to handle these versioning systems, by keeping the most current files available. All you need to do is download them from your team page. Click on the file, and that will take you to its textual domain. Click on the file link, you have a file! ;-)

Automatic update

If you have asked the TP to send you updates to your assigned files automatically, these will simply arrive in your Inbox. You don't need to download them. :-D

Updating is usually quick work, so it's great to have them arrive automatically: a file could be uploaded at the TP with a couple of new or changed strings, sent out to the translator, edited and returned all in the same day. *That's* currency. 8-)

? Other projects have their own howtos on getting current files: ask your team leader.

A new file

You have a clean start: nobody has edited this file before you. ;-)

Edit the headers

as shown above.

Not repeating yourself

The good news, now, is that you don't have to type every single string into that new file, yourself, if you have any compendium files. A compendium is a glossary created by gettext. Your team-leader should be able to point you to current glossaries in whatever form, although we need *compendia* for the command-line process below.

It's best to use the same glossaries as the rest of your team, as a consistent vocabulary is important. It confuses the user much less, and gives him/her less new terms to handle. When you are starting out in computing, or using a new program (we're always learning new things), you don't want to have to worry about differing ways of saying the same thing.

A **compendium** is a text file built by gettext, by merging the contents of completed .po files. You may want to keep different compendia for different types of files: I have different compendia for main program files, games, iso-files and calculator programs. You can apply any number of compendia to a file.

When you apply a compendium to a new file, called *initializing* the file, gettext tries to match the original strings with strings and translations recorded in the compendium. If the match is exact, gettext will fill in the msgstr completely, for you. If the match is close ((in gettext's judgment, and there are debates about how close it needs to be :-))), then it fills in the translated string, but applies the fuzzy tag to that string block. That means: "Check this one, I'm not sure." Even if that string is not completely translated, it may save you time: perhaps a capital letter or punctuation mark is different, or part of the sentence … or it may be completely off-target, but usually it is close, and that's a big help.

! How do we do that? Here is the command (record it somewhere handy):

```
msgmerge --compendium compendium.po -o file.po /dev/null file.pot
```

This says:

Program msgmerge (gettext's merge program), *I want you to use the information in a compendium file, its name* (in this case) *is compendium.po* (it can be anything.po), *I want you to output* (-o) *the combined data from the compendium and the file to a file named file.po, at /dev/null* (because you don't want the combined data, you want the data that matches, /dev/null is like saying, throw it away), *and the file I want you to initialize is called file.pot.*

So, that command could be:

```
msgmerge --compendium glossary1.po -o file.po /dev/null gnubiff.pot
```

Parts of that command:

msgmerge – the program you're asking to do the job

–compendium – the option that says "make a glossary file out of this data"

glossary1.po – the filename of your existing glossary file, or the filename you want for a new one

-o – output the combined two files

file.po – to this file

/dev/null – and lose it, because I don't want the two files combined

gnubiff.pot – but put any matching strings into this file (the one you want to translate)

So all you really need to do is to type the name of your glossary file, your compendium, instead of *compendium.po* here, and type the name of the file you want to translate, instead of *gnubiff.pot*.

! Remember that the path, any directories that msgmerge needs to travel through to find a file, is part of its file-name. The two files in our example might be:

```
Documents/glossaries/glossary1.po
```

and

```
Documents/TP/gnubiff-2.1.3/gnubiff.pot
```

! When typing filenames in the Terminal, use the Tab key to fill in the rest of a name, once you're past any letters that match other names at that level.

Using this msgmerge command may get a lot of matches, or it may not: it depends on how much data you have in your compendium which is relevant to your new file. You can list compendia, one after the other, if you want to apply more than one:

```
msgmerge --compendium glossary1.po glossary2.po glossaryA.po -o file.po /dev/null␣
↪gnubiff.pot
```

Most of all, when you translate a number of files which do similar tasks, or you decide the next time someone asks you to translate the "OK" button, you'll scream and throw things, msgmerge can save you a lot of hassle. It's another of our useful translation tools. (This whole task was very messy before gettext.)

An incomplete file

Firstly, update the headers, as shown above. The version number, translator details and revision date are the key areas when updating.

With an incomplete file, you can use the msgmerge command again: it will simply try to match any strings which are not yet translated.

Before we get down to editing our file, here are a few more time-saving words on building your own compendia.

Our own glossaries

Creating your own glossary files, compendia, is is a simple process, which some of the .po editors have built-in. In LocFactoryEditor, for example, I can create, merge and apply any number of glossaries in various formats (I usually use .tmx).

If using the command line, you can still do it like this, each time you complete a file and want to add its translations to a compendium file:

```
msgcat -o compendium.po file1.po file2.po
```

This command says: *program msgcat* (gettext's catalogue program), *I want you to put all the output* (-o) *from this task in a file called compendium.po.* (If there is already a file with that name in that location, it will merge with it – handy for updating your compendium). *Take all the data from these files: file1.po and file2.po*

so it could be:

```
msgcat -o glossaryA.po gnubiff.po
```

if you are adding only one file to glossaryA, or

```
msgcat -o glossary_kids.po tuxpaint.po gcompris.po
```

if you're adding those two files to your kids' program compendium.

The compendium process is a real time-saver for us, so please take the time to use it. You can always ask for help, or ask questions, on the TP mailing list, as mentioned above.

! I recorded these two commands in a handy place, so whenever I need them, I can copy them in. If you use them often, you may find they stick in your mind. 8-) My mind is not particularly sticky nowadays. More like sludge, I think. :-/

the gettext manual:

invoking the msgmerge program

using translation compendia

Translating a file

You've got the headers sorted out, you've used your compendia to supply any likely strings, and you can't wait to see what weirdnesses our developers have foisted on us now – uh, time to translate. ^_^

Your .po file, apart from the headers, consists entirely of string blocks. Each string block represents one string which will be displayed in translated form in the program from which the .po file was generated. It might be text on a button, on a toolbar, in an error message or tip window, wherever it pops up in the program, it's a string block in our .po file. All God's chillun got string blocks. :-D

Here is the structure of a string block:

```
#.Type: boolean
#.Description
#:../exim4-base.templates.master:4
msgid "Remove undelivered mails in spool directory?"
msgstr ""
```

This is a particularly well-structured string-block, from the Debian Installer translation project. Note the two #. lines: the # and a full stop/period . which denote:

A developer comment

```
#.I am a developer comment. :)
```

Developers can save us a lot of hassle by inserting comments which explain the string, or give instructions on how to format it. Most .po files have no helpful developer comments yet, so this one stands out. You may like to encourage your developers to insert comments, as well as the Report-Msgid-Bugs-To header. 8-)

Here is an absolutely superb example of the developer comment, again from the Debian installer project:

```
#.Type: select
#.Choices
#.Translators beware! the following six strings form a single
#.Choices menu. - Every one of these strings has to fit in a standard
#.80 characters console, as the fancy screen setup takes up some space
#.try to keep below ~71 characters.
#.DO NOT USE commas (,) in Choices translations otherwise
#.this will break the choices shown to users
#:../exim4-config.templates.master:9
msgid "internet site; mail is sent and received directly using SMTP"
msgstr ""
```

You can't go far wrong with that sort of help.

Back to our first example, which still explains the string a lot better than the average .po document:

```
#.Type: boolean
#.Description
#:../exim4-base.templates.master:4
msgid "Remove undelivered mails in spool directory?"
msgstr ""
```

the two developer comment headers tell you:

- The string is a boolean type, i.e., it will have an answer of Yes or No (1 or 0 from the computer's point-of-view).
- The string describes things for the user.

The next line describes where the string fits in in its program. Sometimes these lines can help us understand what the string needs to do, but not often. :-/

While we're on the comments topic, we translators can insert comments, too.

Translator comments

```
# I am a translator comment. ;)
```

! This can be particularly handy when more than one translator works on a file.

In any case, other translators may work on this file in the future, so it's worth inserting a comment if things need to be remembered. Translator comments must be inserted at the very top of the string block, after the gap from the previous block (the "white space"): note the whole line before each quoted translator comment here. They have a # mark then a space: no punctuation mark. Thus, I have often inserted comments like this:

```
# Don't translate this: it's a variable. Đng dch chui này vì là bin.
```

So we might have:

```
# Don't translate this: it's a variable. Đng dch chui này vì là bin.
#. login window data
#:../exim4-base.templates.master:4
msgid "(${NAME})"
msgstr "(${NAME})"
```

or you might suggest a certain way of explaining or formatting something. Don't feel shy about inserting translator comments: they're not seen by the user of the program. You may wonder if some developers know their comments field is meant for talking to us: some programs only contain developer comments where they are talking to each other, even insulting the user. This is disappointing. :-(

! As you work your way through each string block, don't feel that you have to know everything.

Some strings (maybe many of them) will be confusing or even abstruse: many developers do not have good explanatory skills, even in their own language. Feel free to improve the structure, when creating the translated string, and to explain it in a way that will work best for your language group.

! The aim is not to translate the exact word or term, since computing terms are mostly chosen for brevity.

Words like "icon" and "text" were not in general use in the English language before personal computing, so you can choose a brief word or expression which serves to carry the meaning. For example, the word "icon" in Vietnamese is "biu tng", which is considerably longer. Where space is important, in a menu item or on a button, or as the title for a table column, I would use a word for "picture": "hình" or "nh", because they are much the same size as the word "icon", and in that context, where people are expecting a small picture, they carry the appropriate meaning. Computing vocabulary is growing and developing in all languages: you have the opportunity to help create and refine it for your language group.

Most likely your language group will have an ongoing glossary project for computing terms, where you can suggest, find and discuss the appropriate terms. We have one here.

! Your input is important: the aim is to communicate effectively with the user, not to mirror exactly what people are doing in English.

This is even more of a challenge where your culture is very different from the Anglo culture, so give yourself the chance to think carefully about what each string is supposed to achieve, and how to communicate it to your language community.

For example, in Vietnamese, we show emphasis more with the words chosen, than by exclamation marks. Quotation marks interfere with meaning, since we use so many accents, so I use «guillemots» instead. English language to the user from the computer is nearly always wrong for Vietnamese: I need to find the appropriate way to express what the string is really saying. For example:

```
msgid "Choosing a simple root password is a really dumb idea."
```

is insulting in Vietnamese, and completely inappropriate, so my sentence in Vietnamese says something more like:

```
msgstr "It is not a good idea to choose a simple root password."
```

since that form is much stronger in Vietnamese than in English, quite strong enough to gain the user's attention at the right level.

! Remember, while the developer may be the expert on how that program works, you and your team-mates are the ones who understand your language and culture, so *you* need to make the choices about how to express meaning, and the most appropriate way to talk to the user.

Obsolete strings

```
#~ msgid "I am an obsolete string. Nobody loves me. Boo-hoo. :("
#~ msgstr "Tôi là mt chui cũ. Không có ai thng tôi. Hu-hu. :("
```

Strings starting with the hash # and tilde ~.

```
#~ msgid "Forward _Quoted"
#~ msgstr "Chuyn tip _trích dn"
```

Some files will have a number of strings at the end of the file, where the msgid and msgstr string pair start with the hash character, and often the tilde character as well, which signifies the user directory on your hard drive, for example. *It doesn't mean that here.*

! In a .po file, strings starting with #~ are not currently being used by the program.

So why keep them, you may ask? Indeed you may, I've asked the same question myself. These strings may be re-used one day, so you are not advised to delete them. However, you may make your own decision on how much of your energy you are going to devote to these obsolete strings. There is definitely a fault in the process: I've encountered files with nearly all the file obsolete strings!

Your .po editor may keep these strings out of your way. Most PO editors (like Virtaal) will hide them from you.

the gettext manual: obsolete strings

Style tips

In order to save time debugging (removing mistakes from) this file later on, there are several things you need to remember as you progress through the file.

! You must never edit the original string, the msgid.

This information belongs to the program, and if you change it in any way, by so much as a space or moving a word up or down a line, this will cause problems when the file is re-integrated into the original program.

! If there are errors in the msgid, please report them to the developer.

You do this via the Report-Msgid-Bugs-To address in the header, or, if that's not filled in or present, you go to the textual domain for this file, (the page on the TP site from which you downloaded it, linked from your team page) and follow the links to find a contact address. Once you have found it, please fill in the Report-Msgid-Bugs-To header, so no future translator, or you yourself later on, will have to waste time hunting for it again. ;-)

Remember, when you write to the developer, be polite and friendly. It's very easy to get impatient, when you're cleaning up the nth messy .po file, but please remember that these people are also volunteering their time, and may not have great English skills, or even understand how the gettext process works. Make friends: it's a great opportunity. :-D

! Each string must "begin and end with a double quotation mark".

- Many files still have the older structure where each line break means stopping and starting the quotation marks again. This results in:

```
#: ../gedit/gedit-document.c:1964
msgid ""
"The disk where you are trying to save the file has a limitation on file "
"sizes.  Please try saving a smaller file or saving it to a disk that does "
"not have this limitation."
```

This style is now deprecated (not recommended, we're trying to get away from it), so although you must never edit the original strings, you can format the *translation* in the current style: one quotation mark at each end. So, in my file:

```
#: ../gedit/gedit-document.c:1964
msgid ""
"The disk where you are trying to save the file has a limitation on file "
"sizes.  Please try saving a smaller file or saving it to a disk that does "
"not have this limitation."
msgstr "Đĩa đc dùng đ lu tp tin có gii hn v kích thc tp tin.  Hãy lu mt tp tin nh hn␣
↪hoc lu tp tin này vào đĩa không đt ra gii hn trên."
```

As far as I can work out, you can only remove the extra quotation marks where there is no formal line-break (n). Where the n character is present, I've found I have to leave quotation marks at the beginning and end of each line in the string, as formatted in the msgid.

```
# Do not translate the upper-case quoted terms: they are values for the configuration.
↪ Đng dch k thut đã trích dn bng ch hoa vì là giá tr cho cu hình.
#: ../data/gedit.schemas.in.h:77
msgid ""
"Style for the toolbar buttons. Possible values are \"GEDIT_TOOLBAR_SYSTEM\"\n"
"to use the system's default style, \"GEDIT_TOOLBAR_ICONS\" to display icons\n"
"only, \"GEDIT_TOOLBAR_ICONS_AND_TEXT\" to display both icons and text, and\n"
"\"GEDIT_TOOLBAR_ICONS_BOTH_HORIZ\" to display prioritized text beside icons.\n"
"Note that the values are case-sensitive, so make sure they appear exactly as\n"
"mentioned here."
msgstr "Kiu dáng cho nút thanh công c. Giá tr có th là \"GEDIT_TOOLBAR_SYSTEM\"\n"
"cho kiu mc đnh ca h thng, \"GEDIT_TOOLBAR_ICONS\" nu ch hin th các\n"
"biu tng, \"GEDIT_TOOLBAR_ICONS_AND_TEXT\" nu hin c biu tng và ch.\n"
"Và \"GEDIT_TOOLBAR_ICONS_BOTH_HORIZ\" đ hin th ch u tiên cnh biu\n"
"tng. Chú ý là phi vit hoa các giá tr đ đm bo chúng đc hin th\n"
"đúng nh đã nói."
```

Which looks like a multiple shopping-trolley collision. :-/

! Lines ending in a line-break (n) in the msgid must also end with one in the msgstr.

This doesn't mean you have to maintain the same number of lines: you can have more or less lines in the translation than in the msgid. However, any line that had to be broken with a n in the original string, must do the same in the translation. Let's have a look at a few examples:

```
#: ../data/gedit.schemas.in.h:74
msgid ""
"Specifies the number of spaces that should be displayed instead of Tab\n"
"characters."
msgstr "Xác đnh s khong trng đc hin th thay vì ký t Tab."
```

This is correct, because my translation was shorter, so I *didn't* need to break the line.

```
#: ../data/gedit.schemas.in.h:74
msgid ""
"Specifies the number of spaces that should be displayed instead of Tab\n"
"characters."
msgstr "Xác đnh s khong trng đc hin th thay vì ký t Tab, và mt s t thêm na không cn␣
↪thit."
```

This is not correct, because I *did* need to break the first line, as the original did, and I didn't use a n as it did.

So this would be correct:

```
#: ../data/gedit.schemas.in.h:74
msgid ""
"Specifies the number of spaces that should be displayed instead of Tab\n"
"characters."
msgstr "Xác đnh s khong trng đc hin th thay vì ký t Tab, và mt\ns t thêm na không cn␣
↪thit."
```

and even this:

```
#: ../data/gedit.schemas.in.h:74
msgid ""
"Specifies the number of spaces that should be displayed instead of Tab\n"
"characters."
msgstr "Xác đnh s khong trng đc hin th thay vì ký t Tab, và mt\ns t thêm na không cn␣
↪thit. Hn na, tôi có th nói chuyn bng cách\nnày đc my ngày."
```

The result has to be the same layout as the msgid. If it needs to break each line at a certain number of characters (roughly), then you do the same, regardless of how many lines are involved.

You will have noticed the backslash used in the line-break. This is a special character in .po files (and in many others). n means a line-break.

! The other most common use of in .po files is to *escape* quotation marks.

As you will have seen, quotation marks already have a job to do in the string block. They say, *The msgid or msgstr string starts* *"here**, and ends **there."*** So when the gettext parser checks through the .po file, it knows not to try and read what's in between those quotation marks as commands. It gets to loaf off until the next quotation mark tells it that lazy time is over, and it had better pay attention again. :-)

This is all very well, but what if the string itself contains a quotation mark? Oops. . . let's have a look:

```
#:../src/window-commands.c:162
msgid "See the "Quick Help" for a list of commands."
msgstr ""
```

What's going to happen? Well, we know that the parser is going to treat the second quotation mark as the end of the string. Not so good. Then it will try to read everything after that as commands . . . until it hits another quotation mark, which it may think is the beginning of another string. Very messy. You'll see how mixed up it gets in this situation, when you forget a quotation mark or insert an extra one. :-D

Fortunately, we can *escape* this situation, by using the handy backslash. The backslash tells the parser to ignore what these quotation marks normally do. We end up with this, instead:

```
#:../src/window-commands.c:162
msgid "See the \"Quick Help\" for a list of commands."
msgstr ""
```

It looks a bit funny, but it's just a backslash *escaping* each quotation mark. All you need to do is to remember to do that any time you use a quotation mark in your strings, as you might in translating the string I've quoted. Then again, you might use «guillemots», as my language does, and they have no job to do in .po files, so they don't need escaping. So there. ;-)

Another option is to use the curly quote signs Unicode provides: "". they have no special significance either, and look better, at the same time!

! The number and kind of variables in the original and translation must match.

Variables tend to follow certain forms, primarily strftime and printf, but a good general guide is that anything that isn't a piece of normal language is probably a variable. Variables must **not** be changed, because they are placeholders for the program: it has been told, for example, when you see the variable %s in string c:219, it should substitute the user name of the current user. In which case, the string in the .po file:

```
#: src/gbiff2.c:219
#, c-format
msgid "Welcome to gnubiff, %s!\n"
```

when used by the program, will display:

Welcome to gnubiff, Clytie!

if that is my username on that system.

So simply translating it, and leaving the variable where it is, would probably work:

```
#: src/gbiff2.c:219
#, c-format
msgid "Welcome to gnubiff, %s!\n"
msgstr "Chúc mng vào gnubiff, %s!\n"
```

Note that this string breaks the line, although it's quite short. There will be display reasons for this line-break, so we simply do the same.

Although we can copy the language in the string, and the variable…

! You achieve a translation of a much higher quality if you take some time to think about what the string is going to do in the program.

This can be difficult without developer comments explaining the string. However, with a string like this, you will become aware that programs often talk to the user in this anthropomorphic way (cute word, huh? it means 'pretending to behave like people': some of us have had anthropomorphic ex-partners :-D). Where was I? Oh, yeah… um, programs do this "Hi there," stuff, so it's a likely occurrence. In which case, I would do better in my language by eliminating the exclamation mark, which is not appropriate, choosing the verb "using" instead of "entering", and putting the username variable before the implicit verb (using), thus:

```
#: src/gbiff2.c:219
#, c-format
msgid "Welcome to gnubiff, %s!\n"
msgstr "Chúc mng %s dùng gnubiff.\n"
```

Welcome, Clytie, to using gnubiff.

You can change the position of the variable, as I have here, as long as you don't change the **order** of variables. Some strings have more than one variable: a string might say:

```
#: src/gbiff2.c:219
#, c-format
msgid "Welcome to %s, %s!\n"
```

and the program be instructed to fill in first, the name of the current part of the program, and secondly, the username of the current user:

Welcome to gnubiff configuration widget, Clytie!

Since, from the reasons explained above, I would be putting the username variable after "Welcome to (using)", I would be changing the order of the variables:

```
#: src/gbiff2.c:219
#, c-format
msgid "Welcome to %s, %s!\n"
msgstr "Chúc mng %s dùng %s.\n"
```

Welcome, gnubiff configuration widget, to Clytie.

:-X

So I need to indicate the change in order:

```
#: src/gbiff2.c:219
#, c-format
msgid "Welcome to %s, %s!\n"
msgstr "Chúc mng %2$s dùng %1$s.\n"
```

by placing the 2$ (which says 'second variable') and 1$ ('first variable') between the % and s of the variable. This tells the program that variable %2$s might be first in the string, but it's actually the second variable in the program. %1$s might be second, but it's identified as the first variable. The program happily substitutes the current values and I see:

Welcome, Clytie, to using gnubiff configuration widget.

:-)

! So, keep the same number, exact appearance and order of variables in strings. If you need to change the order, use the process above.

Checking your file

If you miss any of these things, or confuse them in any way, do not despair, because when you finish the file (or at any other time), you can run a check on common mistakes, using this command:

```
msgfmt -cv /dev/null FILENAME
```

This says, *program msgfmt, check* (-c) *the language rules (outputting any results to /dev/null because I don't want to keep a copy) in this file.*

msgfmt will list any remaining errors, with line numbers and descriptions, so you can fix them. It will tell you if there are any remaining fuzzy entries, and what types of errors you have. msgfmt is a big help. :-)

Running that check on a file I'm editing now:

```
Pearl:~/gnome/HEAD clytie$ msgfmt -cv gedit/po/vi.po
```

Note that I'm two levels down from my home (user) directory, inside the HEAD folder which is inside the gnome folder, and I need to tell msgfmt that the file vi.po is two levels down from where I am, inside the po folder which is inside the gedit folder. All clear? Hope so. Here we go…

```
Pearl:~/gnome/HEAD clytie$ msgfmt -cv gedit/po/vi.po
gedit/po/vi.po:504: parse error
gedit/po/vi.po:643: missing `msgstr' section
gedit/po/vi.po:644: keyword "t" unknown
gedit/po/vi.po:1385: keyword "C" unknown
gedit/po/vi.po:1386: keyword "C" unknown
gedit/po/vi.po:1402: keyword "C" unknown
gedit/po/vi.po:1403: keyword "C" unknown
gedit/po/vi.po:1409: keyword "C" unknown
gedit/po/vi.po:1468: missing `msgstr' section
gedit/po/vi.po:1469: keyword "n" unknown
gedit/po/vi.po:1483: missing `msgstr' section
gedit/po/vi.po:1484: keyword "ang" unknown
found 12 fatal errors
```

Fatal errors don’t actually kill you, but they will prevent your file from being submitted as complete. Note the helpful line numbers. I’ll have no trouble finding what’s wrong with those: from experience, I’d say I’m missing a few quotation marks, that’s why the parser (a program that reads grammar, in this case the grammar of commands) is trying to read the string as a command, and doesn’t understand the keyword, the first word in the string, as far as a parser is concerned.

You can check your file repeatedly (the up-arrow repeating the last command), until you get a result like this:

```
msgfmt -cv dasher/po/vi.po
133 translated messages.
```

Then you can submit your file. ;-)

Submit your file

In order to submit a completed translation file ((see your team leader for help with any files you can’t complete)), all you need to do is email them to the TP robot program.

! Make sure your msgfmt check comes up clean, with no errors, before sending.

! Make sure the details in the subject line of the email are exact, or your file will not be accepted.

! Make sure you have changed the name of your file to languagecode.po, in my case, **vi.po Note**: you may wish to keep the complete filename, e.g. (in my case, and for the file gnubiff-2.3pre1) **gnubiff-2.3pre1.vi.po** to avoid confusing files with the same name. Another useful precaution is to gzip your file before attaching it to the email: this prevents the encoding being scrambled in transit.

Email address for submitting files:

```
robot@translationproject.org
```

Subject line of the email:

```
PACKAGE_NAME.LANGUAGE_CODE.po
```

For example, with gnubiff in Vietnamese:

```
gnubiff-2.1.3.vi.po
```

! Make sure the package name is exact, a hyphen between the program name and the version number, and full stops/periods in the version number.

! Make sure there is one full stop/period between the version number and the language code, and between the language code and the po extension.

I’ve made a template in my mail program, so whenever I have a file to submit, I only have to fill in the package details. This saves me making mistakes with the rest of it, because it’s easy to slip up on a space or a full stop. You might like to set up something similar. For my email program Mail in Mac OSX, I used Mail Template, an excellent program to save time and trouble in repeated, even reactive mailings.

Where to from here?

I hope you have found this information, which I've scraped together by making probably every conceivable mistake :-D, useful. Please feel free to add to it. I look forward to seeing your experiences here.

? If there is any part of this document which you find hard to understand, please leave a note here, and I will try to explain it.

? We would welcome translations of this document, or any similar howto, in your language.

Enjoy your translating time in the exciting and welcoming Free Software community.

from Clytie

1.10.17 Translating Google

NOTE: It seems that as at 13 Oct 2004 Google says that they "cannot support more languages at this time". So I'm afraid your language will probably not be supported.

While not Open Source or Free Software the Google search engine is a powerful tool used by many on the Internet. Google can be localised for you language using their own custom interface.

How to translate

The process is simple, register and then follow the on screen instructions. One advantage of translating Google is that you can quickly see what your translation will look like in real life. That is an advantage of a web based applications.

1.10.18 Skype Localization

Arguably the most important VoIP solution that can be localised. This Skype devzone article shows the importance of localisation – only 28% of users of Skype make use of an English interface.

How to localise Skype

- Skype Translation forum
- Adding unsupported languages (Windows)

Skype makes use of Qt Qt .ts files for localisation. Thus a compiled Qt .qm file should bring localisation to your Skype GUI. Unfortunately, certainly on the Linux builds, Skype does not seem to detect languages based on the presence of a language file, rather it seems to have a pre-set list of possible interface languages. Rather sad as this would have been an easy GUI to localise based on the power or Qt.

It is possible to update languages that exist within the Skype application. This thread (in Hungarian) links to Hungarian .ts and .qm files. In a Fedora install these are missing and by adding the files Hungarian becomes an available interface languages. This does not work though for Afrikaans (even if the files are renamed), which seems to indicate that the supported languages are pre-set within the binary.

Skype Translation Memory

- Current text based file – 2,900 phrases in 28 languages
- Older spreadsheet with all existing translations (caution most are hidden so make sure you apply a filter again to see the others)

1.10.19 Translating Manpages

Manpages, or manual pages, displayed in your terminal by the **man** program, are an important resource for computer users. They are distributed with the system, and they contain a great deal of immediately-accessible and specific information on how to use software.

`man man` will tell you all about the **man** program, but briefly: you type `man <command>` (e.g. `man find`) to view the manual page about that command or program. Use the **spacebar** to page down, and type **q** to quit (this clears the manpage completely and returns you to the point where you called that page).

Manpages have been translated, and distributed separately by language groups, or made available at certain sites. The aim of a small project liaising between GNU and The Translation Project is to integrate the translation process with the central manpage distribution process. Manpages will be translated, just like applications, then distributed with the original package, just like applications.

The manpage format, **(g)roff** or similar, is not easy to translate. However, the po4a conversion filters remove this problem completely, by giving us the capacity to convert manpages to our familiar PO format. (See more information on conversion filters you can use in *Non-PO formats*.)

Pootle has integrated po4a and the Translate Toolkit filters, which enabled me recently to upload a trial manpage to Pootle, have it automatically converted there to PO format, translate it easily, then have it converted back to g(roff). This simplified the translation process considerably.

Debian has just made a package of manpages available in PO format, again using po4a. This is a big step forward, and will result in many more, and more current, manpage translations.

I found, once I’d translated my pilot manpage (find1), that I had to do some configuration before my UTF-8 (Vietnamese) manpage would display correctly in my terminal.

XXX

Manpage process

- Get translated manpage(s), place in MANPATH
  - If you’re unsure what your MANPATH is, type `echo $MANPATH` — this shows all the places the **man** program will look for manpages.
  - Create a folder/directory named for your ISO language code (e.g. for Vietnamese, I create a folder called **vi** within my MANPATH : I placed it in `</usr/share/man>`.
  - Create a sub-folder (`manX`) for each number shown after a manpage name. E.g. because I have the **find1** manpage, I need to create a **man1** sub-directory, which for me was `</usr/share/man/vi/man1>`. Place each manpage in the appropriately-numbered sub-directory.
- Install the latest version of groff, if you don’t have it. You can get it via **fink**, or your usual package manager.
- Install groff-utf8. Bruno Haible is currently working on a full release of groff, which will support UTF-8; meanwhile, he has created this patch for us. Thanks, Bruno! :-)
- Set your **locale**, if it has not yet been set.
  - Your locale tells your computer what language you prefer to use, plus some other useful information related to your language choice.
  - You type your locale into your shell profile. (FIXME Does this differ from one shell to another?)
  - For **bash**, find and edit your `<~/.bash_profile>` or `<~/.profile>` file, where ~ is your username, your user directory. You can edit this type of file in your text editor. I used BBEdit for Mac OSX.

  - Type `export LC_ALL=ll_LL.UTF-8` where `ll_LL` is your language code (and possibly country), e.g. for me it was `export LC_ALL=vi_VI.UTF-8`.
  - Save the file, don't change the name or location.
- Find the file </etc/manpath.config> or <usr/share/misc/man.conf>. If you don't have either of these, it will be something similar. **man** has to get its config. info somewhere!
- Copy this file to <~/.man.conf>. (This leaves your original man config untouched, if you want to return to it later, or if the machine has multiple users.)
- In that file, find the **NROFF** line, something like this:

  ```
  ``NROFF            /usr/bin/groff -Wall -mtty-char -Tascii -mandoc -c``
  ```

  and edit it to:

  ```
  ``NROFF            /usr/bin/groff-utf8 -Tutf8 -mandoc -R``
  ```

  - Make sure you have a terminal (monospaced) font set which supports your language *and supports UTF-8*.
  - In your terminal, call your translated manpage!
    * All I had to do was type `man find`. Because I had set my system to call documents and software in my language, and because this manpage was available, translated, in the directory for my language code, **man** automatically showed that page. It displayed beautifully (or as beautifully as my language can in a monospaced font ;-)).

Please add any information specific to your system or language. This information will help ensure an effective manpage translation and distribution process. :-)

Clytie

<clytie@riverland.net.au>

- Desktop Systems
  - *GNOME*
  - *KDE*
  - *XFCE Translation*
- Major Applications
  - *Mozilla Product Localisation*: *Firefox*, *Thunderbird*
  - *OpenOffice.org*
- Distributions
  - *Debian Translation*
  - *Translating Fedora*/Red Hat
  - *Mandriva*
  - *SUSE*
- *Wikis*
- *The Translation Project* (some parts need to move to other sections)
- Other
  - *Translating Google*

   - *Skype Localization*
   - *Man pages*
   - WINE
   - ReactOS – Microsoft Windows clone
   - OpenMoko

1.11 Translation

1.11.1 Translation Guidelines

What is the policy for translating acronyms in my language? Should I use Title Case? These are all issues that should be addressed in a translation guidelines or policy document. Some of these already exist but are usually project specific. This page and template are a space for you to develop a policy document for your language.

Guidelines Template

Translation Guidelines (Template)

This is a template, use it to start your own guidelines document. Also update it with improvements you make in your document.

Capitalisation

Title Case

Title Case if where you make the first letter of each word capital except words like: is, are, a, an.

> This is Title Case This is sentence case

If you see something in title case do you change it so sentence case?

Plurals

Gettext Plural Header

What should your gettext plural header be set to?

English style text based plurals

English style plurals are where the text uses both the singular and plural:

> Please enter the author(s) name. Document(s)

In some languages if you had to use the same bracketed construct it looks ugly. In Nguni based languages we convert such sentences to only use the plural form, as we feel the singular is inferred and using the bracketed construct in ugly.

Punctuation

French Style

Some sentence appear with spaces before end punctuation:

> I am a french style sentence ! Are you sure ?

We call this French style punctuation, should translators keep this style or convert it to a normal style?

Brand Names

Should you translate brandnames?

Do not translate these brands

- OpenOffice.org * **-** OpenOffice.org, Writer, Calc, Impress * **+** Math, Chart

Where **+** means yes and **-** means no. You might want to supply the valid translations for all of these.

Glossaries

List the glossaries that the translator should use.

The *Translation Guidelines (Template)* is designed to be a basis from which to develop your languages guidelines. If you see that you re adding a new section to your policy first check to see if its in the template. If not then please add it. We want the template to be a model of what issues people should be considering. The policy document is yours – make it do what you need it to do.

Language Guideline Documents

Translation of specific phrases in Afrikaans

unable to

Unable to (v) = Kan nie (v) nie
Unable to (v) (n) = Kan nie (n) (v) nie

error

Error (v)ing (n) = Kon nie (n) (v) nie
Error occurred (v)ing (n) = Kon nie (n) (v) nie
Error = Fout
Error occurred = 'n Fout het voorgekom
An error occurred while = Kon nie . . . volledig . . . nie
There were errors (v)ing (n) = (n) kon nie volledig (v) nie
There were errors (v)ing (n) = Kon nie (n) volledig (v) nie
(n) error = (n)fout
(v) error = (v)fout

Error in (n) = Fout in (n)
returned an error = 'n fout aangegee

German Translation Guidelines

(Richtlinien für die Deutsche Übersetzung)

Dieses Dokument enthält die Richtlinien die speziell bei deutschen Übersetzungen zu beachten sind. Bitte lesen Sie auch die *allgemeine Einführung*, da hier nur auf Besonderheiten der deutschen Sprache eingegangen wird.

Empfehlenswert ist auch das Handbuch für Übersetzer des deutschen KDE-Übersetzungsteams (speziell das Kapitel „Inaltliches" und die nachfolgenden Kapitel, aber auch etwas technischer Hintergrund (speziell Gettext und PO-Dateien) kann nicht Schaden).

Groß-/Klein-Schreibung

Im Gegensatz zum Englischen werden im Deutschen nicht nur Wörter am Satzanfang, Eigen- und Ländernamen sondern außerdem auch alle Substantive (Hauptwörter) groß geschrieben.

Titel

Handelt es sich um einen Titel (Buch, Film, usw.) werden im Englischen fast alle Wörter groß geschrieben, dies ist im Deutschen nicht so, hier wird normale Groß-/Kleinschreibung verwendet.

Dances With Wolves -> Der mit dem Wolf tanzt

Getrennt-Schreibung

Im Englischen werden viele zusammengesetzte Hauptwörter dadurch gebildet, dass man die Wörter einzeln hintereinander schreibt. Im Deutschen hingegen wird entweder ein Bindestrich verwendet oder das Wort zusammengeschrieben:

localisation guide -> Übersetzungsleitfaden oder Übersetzer-Leitfaden

SS und ß

Entgegen manchen Gerüchten gibt es auch nach der Rechtschreibreform noch das ß. Es kommt dann zum Einsatz wenn ein scharfes S auf einen langen Vokal oder mehrere Vokale folgt:

<del>Strasse</del> Straße; <del>Massband</del> Maßband aber Knetmasse
<del>Weiss</del> Weiß; <del>aussen</del> außen

Apostrophe

Im Deutschen wird das Genitiv-s nicht mit einem Apostroph vom zugehörigen Wort getrennt wie das im Englischen der Fall ist. Weder im Deutschen noch im Englischen wird das Plural-s mit einem Apostroph abgetrennt. Generell sollte man den Apostroph nur sparsam verwenden.

David's car -> Davids Auto
<del>Apple's</del> Apples are red

Plural (Mehrzahl)

Gettext Plural Header

```
Plural-Forms: nplurals=2; plural=n != 1;
```

Hartcodierter Plural

Manchmal wurde der Plural in Klammern in den Text eingefügt anstatt zwei Texte zu verwenden wie Gettext es eigentlich vorsieht. Da ein Plural im Deutschen eine ganze Reihe weiterer Änderungen im Satz nach sich ziehen kann und auch in der Regel nicht so leicht wie im Englischen mit einem angehängten s gebildet wird sollte man wenn der Aufwand (und Platzverbrauch) zu groß wird die Mehrzahl verwenden.

Please enter the author(s) name. -> <del>Bitte geben Sie den/die Namen des/der Autors/Autoren ein.</del> Bitte geben Sie die Namen der Autoren ein.
Document(s) -> Dokument(e)

Höflichkeit

Es sollte grundsätzlich die formale Anrede (Sie) benutzt werden, dabei ist darauf zu achten, dass Sie, Ihr, Ihnen usw. wenn es sich um eine Höflichkeitsform handelt groß zu schreiben ist, wenn es sich jedoch um etwas anderes handelt (zum Beispiel Bezug zu anderen Personen als dem angesprochenen Benutzer) klein.

Die informelle Anrede (Du) sollte nur in sehr seltenen Fällen (z.B. Computerspiele die ausschließlich auf ein jugendliches Publikum ausgerichtet sind) benutzt werden, da sie leicht sehr künstlich (auf jung getrimmt) wirkt.

Translate this programme into your language -> Übersetzen Sie dieses Programm in Ihre Sprache

Weibliche Formen

Wo immer möglich sollte man versuchen geschlechtsneutrale Bezeichnungen zu verwenden. Ist dies nicht möglich so empfehle ich den generischen Maskulin, da man sonst Gefahr läuft dass die Texte zu lang und schwer lesbar werden.

students -> Studierende (statt Studenten und Studentinnen)
the user -> der Benutzer (statt der/die Benutzer/-in)

Zeichensetzung

Die deutsche Zeichensetzung unterscheidet sich (bis auf die Kommasetzung) nicht wesentlich vom Englischen. Insbesondere bei Sonderzeichen (alles außer Punkt und Komma) sollte peinlichst darauf geachtet werden, dass die Zeichensetzung bestehen bleibt. Kommas werden im Deutschen allerdings wesentlich häufiger verwendet als im Englischen, wo sie eigentlich nur dazu dienen in Situationen die sonst mehrdeutig wären Klarheit zu schaffen. Auch sollte man bei Sätzen die durch die Übersetzung zu lang werden nicht davor zurückschrecken sie in zwei separate Sätze aufzutrennen.

Vorsicht bei auf den ersten Blick unsinnigen Zeichenfolgen!

Oftmals handelt es sich dabei um spezielle Markierungen zur Formatierung des Textes. So werden z.B. in vielen Programmiersprachen Zeilenumbrüche, Tabulatoren und mehrere aufeinander folgende Leerzeichen (diese Zeichengruppe wird als „Whitespaces“ bezeichnet) so behandelt als stünde an ihrer Stelle ein einziges Leerzeichen. Will man also eines dieser Zeichen im ausgegebenen Text erzeugen muss man sogenannte Escape-Sequenzen benutzen (z.B. `\n` für einen Zeilenumbruch, `\t` für einen Tabulator, `\\` für einen Backslash (in C-verwandten Sprachen, HTML hat wiederum andere Escape-Sequenzen)). Auch Links werden oft speziell (z.B. durch eine zweifache eckige Klammer) gekennzeichnet um sie im Programm anklickbar zu machen. An dieser Stelle sei nochmals auf die *allgemeine Einführung* verwiesen.

Es ist also überaus ratsam Zeichensetzungen exakt aus dem Original zu übernehmen, Pootle bietet hierfür die Schaltfläche „Kopieren“ die den Originaltext in das Textfeld kopiert, in welchem dann nur noch der pure Text übersetzt werden muss, so ist garantiert, dass die Zeichensetzung erhalten bleibt.

Recent news:n
Our project has been <nowiki>‘slashdotted <http://slashdot.org>‘_</nowiki>.

Neuigkeiten:n
Unser Projekt wurde auf <nowiki>‘slashdot.org <http://slashdot.org>‘_</nowiki> erwähnt.

Anführungszeichen

Die korrekten deutschen Anführungszeichen sind folgende: „ “ bzw. ‚ ‘ (Regel 9966) und auch Französische sind zugelassen » « bzw. › ‹

Leider lassen sich diese Zeichen nicht so einfach mit der Tastatur erzeugen **(bitte benutzen Sie keine Kommas, größer als, kleiner als oder andere ähnlich aussehende Zeichen, da sie nur zu Verwirrung führen** (z.B. für Blinde die den Text durch ein Programm vorgelesen bekommen oder Leute die andere Schriftarten verwenden als Sie)). Theoretisch kann man die Zeichen einfach per Kopieren & Einfügen aus der Zeichentabelle oder dem obigen Text verwenden oder den Unicode per Tastenkombination eingeben (Windows: Alt-Taste gedrückt halten und Code auf dem Ziffernblock eingeben, Gnome: STRG+Shift+U drücken und dann den Code in hexadezimaler Schreibweise eingeben) dies ist jedoch unpraktikabel.

Linux-Benutzer können stattdessen die Compose-Taste benutzen oder sich mit `xmodmap` die Tastaturbelegung entsprechend anpassen (ich verwende Caps-Lock als Compose-Taste, da sie sonst sowieso keinen Zweck erfüllt). Wem das immernoch zu umständlich ist, der kann auch die „ganz normalen“ Anführungszeichen ” und ‘ (Shift+2, Shift+#) benutzen.

Achtung: „normale Anführungszeichen“ (“) müssen in PO-Dateien mit der Escape-Sequenz `\"` codiert werden, sonst meldet Gettext einen Syntax-Fehler. Pootle und die meisten anderen PO-Editoren machen das zwar automatisch aber man sollte lieber vorher nochmal im Handbuch nachschauen bevor man alles von Hand korrigieren muss.

Die Französische Art

In Frankreich ist es üblich Leerzeichen vor die Satzzeichen zu setzen. Dies ist im Deutschen falsch (und äußerst unschön) und wird als *Plenken* bezeichnet. Auch das Gegenstück, das verzichten auf Leerzeichen um die Satzzeichen ist falsch (soweit ich weiß in allen Sprachen) und wird *Klempen* genannt.

Richtig: kein Leerzeichen vor den Satzzeichen und ein Leerzeichen dahinter (Ausnahmen: öffnende Klammern und Gedankenstriche).

Plenken , also das Einfügen eines Leerzeichens zwischen Wort und nachfolgendem Satzzeichen , ist uncool !
Klempen,der beinahe völlige Verzicht auf Leerzeichen,aber auch!Auch wenn man dadurch Platz spart.

Markennamen

Werden üblicherweise nicht übersetzt. Ausnahmen sollten nur dann gemacht werden wenn die Marke auch im allgemeinen Sprachgebrauch in der übersetzten Form benutzt wird:

Sellotape™ [Br.] / Scotch ® Tape [Am.] -> Tesa {n} (ugs. für Tesafilm®) / Tixo [österreichisch]
Kleenex ® (umg.) (Taschentuch) -> Tempo ® (umg.) (Taschentuch)

Fehler im Original

Sollten nicht einfach in der Übersetzung behoben werden. Man sollte vorerst den Text so übersetzen wie er da steht, die Übersetzung als fraglich markieren und den vermeintlichen Fehler in der Anmerkung darlegen. Dann sollte man die Entwickler kontaktieren und sie auf den augenscheinlichen Fehler aufmerksam machen (die Kontaktaufnahme kann in der Regel am schnellsten über das IRC-Netzwerk erfolgen). Diese Vorgehensweise hat die Vorteile, dass erstens alle Sprachen von der Fehlerkorrektur profitieren, zweitens der eigentliche Fehler behoben wird und nicht nur seine Auswirkungen minimiert und drittens dass für den Fall dass es gar kein Fehler war die Übersetzung korrekt bleibt.

Fragliche Wörter & Anmerkungen

Wenn man sich bei einer Übersetzung nicht 100% sicher ist sollte man nicht zögern diese als fraglich zu markieren und evtl. eine Bemerkung dazu zu schreiben warum man sich nicht sicher ist. Es ist wesentlich besser einige korrekte Übersetzungen als fraglich zu markieren als eine falsche Übersetzung unmarkiert zu lassen. Der Grund ist einfach: während fragliche Übersetzungen häufig nochmals von anderen Übersetzern geprüft werden, kann es sein dass die unmarkierten von keinem mehr überprüft werden.

Auch an anderen Stellen sollte nicht mit Anmerkungen gespart werden, gerade an solchen Stellen an denen auch andere Übersetzungen potenziell in Frage kommen oder wenn man andere Übersetzer korrigiert sollte man kurz begründen warum man diese Form wählt und nicht die andere, auch um zu verhindern, dass beide Übersetzer die Übersetzungen immer wieder gegenseitig korrigieren was schnell zu Streitigkeiten führen kann.

Glossare

Kann ich keine empfehlen (da ich keine benutze). Im Zweifelsfall einfach mal bei einer anderen Anwendung nachgucken. Auch Wikipedia hilft oftmals bei der Suche nach dem passenden Begriff weiter.

Pootle listet bei der Übersetzung möglicherweise relevante Wörter mit ihren Übersetzungen aus dem Terminologie-Projekt rechts auf. Oftmals wurden dort alternative Übersetzungen als Anmerkung hinterlegt welche als Tooltip verfügbar sind (den Mauszeiger für eine kurze Zeit über dem Wort positionieren).

Wörterbücher

Wer ein Online-Wörterbuch sucht ist bei leo.org ganz gut aufgehoben (Vorsicht bei Google-Sprachtools & Co., da sie oftmals nur eine Übersetzung für das Wort auflisten und nicht wie leo.org auch alternative Bedeutungen)

„Analoge" Wörterbücher sind für die Übersetzung von Anwendungen meist eher weniger geeignet, da Online-Wörterbücher oftmals einen größeren Wortschatz in Richtung EDV aufweisen. Auch hier sollte man auf einen ausreichend großen Wortschatz achten und auch alternative Bedeutungen eines Wortes durchlesen.

Hebrew Translation Guidelines

Everyone, use the Carmel word list. It will ease your job.

- Afrikaans * Translation of *specific phrases*
- *German*
- *Hebrew*
- Ndebele, South
- Northern Sotho
- Sotho, Southern
- Swati
- Tsonga
- Tswana
- Venda
- Xhosa
- Zulu

1.11.2 Defining Words

Even for those translators with a good technical understanding of computer terminology and jargon there are some words that they will not understand. Then of course those that they do understand they might not grasp well enough in order to create a translated term. For this you need access to a good dictionary. Fortunately there are a number of good resource available.

Online Dictionaries

- Dict.org – Online dictionary with many resource: Webster, computer terms, etc
- Foldoc (Free On-line Dictionary of Computing)
- Wiktionary – sister of Wikipedia

Online Encyclopedias

- Wikipedia

1.11.3 Common Translation Errors

Translators – even professional translators – usually make these error when translating computer software.

Overview of common errors

Capitalisation

Wrong: no capitalisation in translation
Correct: translation follows translation language capitalisation
convention

Double words

Translator could not decide which word was more appropriate

Wrong: two translated words for one original word
Correct: choose one word if you have two options

Translated things that should stay in English

Wrong: translating program function name like get_file_attributes()
Correct: leave program syntax or function names untranslated

Variables

Wrong: variables left out of translation
Correct: variable in translation in the correct order for the language

Wrong: environment variable names (eg. EDITOR) and possible fixed values
(eg. COLOR, TRANSPARENT) translated
Correct: left in English and perhaps surrounded by single quotes to
indicate that they are not in the target language on purpose.

Punctuation

Wrong: leaving out end punctuation and whitespace (eg. `"File:  "`)
Correct: copying end punctuation and whitespace almost exactly as the
original into the translation.

Wrong: adding missing fullstops to translated sentence
Correct: if full sentences do not contain a fullstop in the original do
not add one to the translation.

Wrong: adding exclamation marks (!) to the translation when they occur in the original
Correct: use exclamation marks to correspond with the tone of the
application and the convention of the translated language. Ie you may leave
them out of the translation.

Wrong: leaving out ellipses (. . .)
Correct: always add ellipses to the translation

Accelerators

Wrong: leaving out accelerators (either _&~ depending on the application)
Correct: if the original has an accelerator so should the translation

Wrong: placing the accelerator exactly in the same position as the original
Correct: place the accelerator on the word / syllable / part of the
sentence that is accented in its pronunciation or is the focus of the sentence.

Wrong: using the same letter for accelerator keys
Correct: try to vary the letters chosen as the accelerator key (in some
languages almost all words start with U or I. In this case make an effort to
choose other letters)

HTML

Wrong: HTML tags are translated
Correct: HTML tags are not translated

Wrong: translatable HTML entities are not translated
Correct: translate items that will be viewable. E.g. HTML img tag's alt
attribute <img alt="translate me">, a tag's title attribute <a href=blah
title="translate me">.

1.11.4 Common Translation Errors

The following are common errors detected by the pofilter tool. It is useful to see these errors in order that you don't commit them yourself. We have given examples of some false positives also.

Double Spacing

Notice the double space in "lefoko la".

```
#: NoColorError
#, fuzzy
#_ doublespacing: checks for bad double-spaces by comparing to original
msgid "Click on a color or enter a valid HTML color string"
msgstr ""
"tobetsa mmala kgotsa tsenya lefoko  la mmala la html le le letleletsweng"
```

The following is a false positive, notice that the double space ("account. There") is in the msgid.

```
#: accountSettingsDesc.label
#, fuzzy
#_ doublespacing: checks for bad double-spaces by comparing to original
msgid ""
"The following is a special account.  There are no identities associated with "
"it."
msgstr ""
"Se se latelang ke akhaoto e kgetegileng. Ga go kitsiso e kamano le yona."
```

End Punctuation

The sentence is missing a fullstop.

```
#: MissingSiteNameError
#, fuzzy
#_ endpunc: checks whether punctuation at the end of the strings match
msgid "Please enter a name for this publishing site."
msgstr "ka kopa tsenyetsa letlakala le la phatlalatso leina"
```

The sentence is missing an ellipsis (. . .)

```
#: AdvancedProperties
#, fuzzy
#_ endpunc: checks whether punctuation at the end of the strings match
msgid "Advanced Properties..."
msgstr "ditiro tsa maemo a kwa godim"
```

Missing colon (:)

```
#: JobTitle.label
#, fuzzy
#_ endpunc: checks whether punctuation at the end of the strings match
msgid "Title:"
msgstr "Setlha"
```

The end contains a space before the colon which shouldn't be there.

```
#: addressMessageTo.label
#, fuzzy
#_ endpunc: checks whether punctuation at the end of the strings match
msgid "Address message to:"
msgstr "Attresetsa molaetsa go :"
```

Notice that the msgid is incorrect (it only has 2 dors) and the msgstr has been corrected. Leave this as it is as the msgstr is now correct. Also report this as an error against the application.

```
#: saveAll.label
#, fuzzy
#_ endpunc: checks whether punctuation at the end of the strings match
msgid "Save All.."
msgstr "Boloka tsotlhe..."
```

There should be a space before "->"

```
#: fieldMapExport.add
#, fuzzy
#_ endpunc: checks whether punctuation at the end of the strings match
msgid "Add Field ->"
msgstr "Atisa lebala->"
```

The msgstr should end in ? but it ends in .

```
#: PKCS12PasswordInvalid
#, fuzzy
#_ endpunc: checks whether punctuation at the end of the strings match
msgid ""
"Could not decode PKCS #12 file.  Perhaps the password you entered was "
"incorrect?"
msgstr ""
"Ga ya kgona go sirolola PKCS #12 file.  Gongwe lefoko-phetiso leo o le "
"tsentseng le fosagetse."
```

The closing bracket should be a round bracket not a curly bracket.

```
#: UnknownCertIssuer
#, fuzzy
#_ endpunc: checks whether punctuation at the end of the strings match
msgid "(Unknown Issuer)"
msgstr "(Mofi ga e itsewe}"
```

End Whitespace

Remove the extra whitespace at the end.

```
#: deleteCardCmd.label
#, fuzzy
#_ endpunc: checks whether punctuation at the end of the strings match
#_ endwhitespace: checks whether whitespace at the end of the strings matches
msgid "Delete Card"
msgstr "Ntsha karata "
```

Start Punctuation

False positive. The translation is correct, although you might want to define a policy on how you handle this form of plural.

```
#: SuccessfulP12Backup
#, fuzzy
#_ endpunc: checks whether punctuation at the end of the strings match
#_ startpunc: checks whether punctuation at the beginning of the strings match
msgid "Successfully backed up your security certificate(s) and private key(s)."
msgstr ""
"(Di/Se)tifikeiti tsa gago tsa pabalelo le (di/se)lotlolo (t)sa bosephiri di/"
"se bolokilwe thuso-morago ka tswelelopele."
```

Start Whitespace

The space at the beginning of the message should be deleted.

```
#: DisableDlgTitle
#, fuzzy
#_ startwhitespace: checks whether whitespace at the beginning of the strings matches
#_ startpunc: checks whether punctuation at the beginning of the strings match
msgid "%S Quick Launch"
msgstr " %S Quick Launch"
```

The space at the beginning of the message should be deleted.

```
#: ko
#, fuzzy
#_ startwhitespace: checks whether whitespace at the beginning of the strings matches
#_ startpunc: checks whether punctuation at the beginning of the strings match
msgid "Korean"
msgstr " SeKorea"
```

Numbers

The number 1252 does not appear in the translation.

```
#: windows-1252.title
#, fuzzy
#_ numbers: checks whether numbers of various forms are consistent between the two␣
↪strings
msgid "Western (Windows-1252)"
msgstr "Turkish (Windows-1254)"
```

Might not be a problem if 2 has been written in full

```
#: SSL2Disabled
#, fuzzy
#_ endpunc: checks whether punctuation at the end of the strings match
#_ numbers: checks whether numbers of various forms are consistent between the two␣
↪strings
msgid "You cannot connect to %S because SSL version 2 is disabled."
msgstr "Ga o kgone go lomagana go %S ka gonne SSL e ntshitswe bokgoni.."
```

Variables

&vendorShortName; should not have been translated.

```
#: throbber.tooltip
msgid "Go to the &vendorShortName; home page"
msgstr "E ya ko &vendorKortNaam; go letlakala la le gae"
```

&quot; should not have been translated

```
#: incomingServerNameDesc.label
#, fuzzy
#_ variables: checks whether variables of various forms are consistent between the␣
↪two strings
msgid "Enter the name of your incoming server (for example, &qout;mail.example.net&
↪quot;)."
msgstr "Tsenya leina la moridi~Zi wa gago yo a go romelago molaet~Za (ka mohlala, &
↪tsopolo;poso.mohlala.net&tsopolo;)."
```

Punctuation Spacing

There should be a space after , in "Ka sekai,netscape". A single quote is also missing.

```
#: abbreviateOn.label
#, fuzzy
#_ puncspacing: checks for bad spacing after punctuation
#_ endpunc: checks whether punctuation at the end of the strings match
msgid "Full names (For example, 'netscape.public.mozilla.mail-news')"
msgstr "Maina ka botlalo (Ka sekai,netscape.public.mozilla.posol-dikgang')"
```

Space missing after colon.

```
#: unreadMsgStatus
#, fuzzy
#_ puncspacing: checks for bad spacing after punctuation
msgid "Unread: %S"
msgstr "Ga ja balwa:%S"
```

Space missing after comma "kenna,o sutlhe"

```
#: defaultcharactersetBidiCmd.label
#, fuzzy
#_ puncspacing: checks for bad spacing after punctuation
msgid ""
"Use my default character set, overriding the document-specified character set"
msgstr ""
"dirisa ditlhaka jaaka ditlhophilwe kenna,o sutlhe tse di tlhophetsweng "
"tokomane e"
```

Space missing after semi-colon "da dk;faele". Also note missing minus between da and DK.

```
# LOCALIZATION NOTE GROUP : DO not localize the entities below; test case
#: da-DK-file.label
#, fuzzy
#_ puncspacing: checks for bad spacing after punctuation
msgid "Danish (da-DK; file)"
msgstr "danish (da dk;faele)"
```

Short

There is a missing sentence

```
#: SIClueless
#, fuzzy
#_ endpunc: checks whether punctuation at the end of the strings match
#_ endwhitespace: checks whether whitespace at the end of the strings matches
msgid ""
"There are unknown problems with this digital signature. You should not trust "
"the validity of this message until you verify its contents with the sender."
msgstr "Gona le mathata ao a sa itsiweng ka mosaeno o wa dinomoro. "
```

Long

The translation looks too long, it might be right but it is unlikely.

```
#: directionBidiMenu.label
#, fuzzy
#_ endpunc: checks whether punctuation at the end of the strings match
msgid "Default Direction"
msgstr ""
"tshupetso e e diragadiwang fela fa ditshupetso tse dirulagantsweng go "
"diragalapele sitwa godiragala."
```

```
#: mediaEncryption
#, fuzzy
#_ endpunc: checks whether punctuation at the end of the strings match
msgid "Encryption:"
msgstr "fetolelo go mokwalo o o fitlhang tshedimosetso ka ga lefoko la tlwaelo"
```

Unchanged

The english has not been translated. The word plugin is translatable. In fact this example show two more errors, the start capital is missing and the colon is also missing. So this translators took an English string and replace it with an untranslated and badly formatted string.

```
#: mediaPlugin
#, fuzzy
#_ endpunc: checks whether punctuation at the end of the strings match
msgid "Plugin:"
msgstr "plugin"
```

URLs and Emails

This is a well translated email address in this case it was used as an example so is translated

```
#: emailExample.label
#, fuzzy
#_ endpunc: checks whether punctuation at the end of the strings match
msgid "(for example: user@example.net)."
msgstr "(ka mohlala: modirisi@mohlala.net)"
```

1.11.5 Accelerator Keys

An *accelerator key* is a key on your keyboard that you can press to quickly access a menu or function. It is also sometimes called a *hot key*, *access key* or *mnemonic*.

What is an accelerator key and an accelerator marker

If you look at the menu bar on any application you will see that the first letter of each entry in underlined. Notice that File, Edit and View each have the first letter underlined. To quickly open the File menu press `Alt+F`. You will notice that most of the menu entries also have accelerator keys, to access Open simply type `O` after typing `Alt+F`.

An *accelerator marker* is the special character we use to mark accelerator keys when we translate.

Identifying the accelerator key

In translations accelerator keys are shown by various characters:

Application	Marker	Name	Source Text Example	Displays As	Note
KDE	&	ampersand	Save &As…	Save As…	
GNOME	_	underscore	Save _As…	Save As…	
OpenOffice.org	~	tilde	Save ~As…	Save As…	
Mozilla	&	ampersand	Save &As…	Save As…	Using moz2po
Windows Windows RC files	&	ampersand	Save &As…	Save As…	

In all of the above examples pressing `A` would take you to the *Save As* dialogue box.

What happens if you want to use the & character without making it an accelerator? In this case the accelerator is usually escaped by using && (for KDE) or &amp; (for Mozilla). E.g. "Mail && News" or "Mail &amp; News". As a translator you would be free to drop the & and use the equivalent of "and" in your language.

Selecting

How do you select an accelerator key for your languages? Keep these principles in mind:

1. Try to keep it similar to the English. This makes it easier for users to switch to localised interfaces.
2. Accelerators should not degrade readability too much.
3. Avoid duplication. Ideally, in any menu, window or dialogue box, a letter should only be used as an accelerator for one string. If the letter 'T' is used for two menu entries, it is usually not a big problem, but the software isn't as usable as it could be.
4. Review accelerators in the running application.

We usually follow these simple rules.

1. Try to keep the same position as the original. E.g. "&File" –> "&Ifayile"
2. Try to keep the same letter as the original. E.g. "&File" –> "I&fayile"
3. Select an uncommon letter. In Xhosa the letter I occurs frequently so we try to avoid using it too much.
4. Avoid adding accelerators under characters with descenders such as lower case g, p, q or y. The character goes below the base line and could cause the character to look odd and the accelerator not to be immediately recognisable. E.g. in KDE &g would appear as *g* which looks messy. It is best to confirm with an actual test, since some rendering systems actually support underlining clearly when the letter contains a descender.

5. Try to avoid letter next to a descender for similar reasons as above.
6. Avoid narrow letters like i and l.
7. Make sure the character is actually available on the user's keyboard. Accented characters such as ä might or might not be present on a keyboard in the language.
8. If your language uses diacritics under certain letters such as ţ, or , avoid these characters as well, as the underline might run through the diacritic. Avoid letters with diacritics entirely unless you are very sure that they will be usable by the end-users.
9. No available character. Usually in Asian languages where a keyboard is Western. Translate the word into your language and use the original English accelerator. Ie "&File" –> "XXXX(&F)". In Mozilla you can safely ignore defining the accelerator in which case it will default to the English version (FIXME: validate this :)
10. Non-letters like numbers and punctuation marks can be used if they are present on a standard keyboard as is common amongst the speakers of your language. This will usually only be a relevant choice if it was chosen for the source text as well.

Accelerator Clashes

If we have two accelerator keys using the same letter then we say that we have a clash. Assume that the following list are the top three entries in a menu:

- Author
- Address
- Available Ations…

You will notice that all the accelerators use the `A` key – this is a clash. Fortunately most applications will cycle through the options as you press A repeatedly. But what would happen if the 3rd item appeared first? (The third item uses the ellipses (…) to indicate that a dialogue box will open. This if this appeared first you would never be able to access the other items.

Here is a better choice of accelerators for the same menu:

- Author
- Address
- Available Actions…

We now use `A`, `d` and `c` – there are no conflicts.

Examples

English	Bad	Good	Why?
X Axis…	Khona ya X…	Khona ya X…	This accelerator appears nicely on the letter X in the English. The translator slavishly followed the first rule of keeping the accelerator in the same position. Yet the good version, which follows rule 2, is much better as it keeps the same letter and it also works for the next string which is //"__Y__ Axis…"//
800x600 pixels	dikarolwana tse 800x600	dikarolwana tse 800x600	This follows the previous example. The translator has continued to use the first letter of the translation even though the number work well in the translation so there was no need to change the accelerator. It also work with the surrounding translations that cover 1024x768, etc. With the option chosen by the translator all of these translations would use the __d__ as the accelerator which wouldn't work.
URL for Perl scripts	URL bakeng sa maqephe a Perl	URL bakeng sa maqephe a Perl	You will notice that Perl is untranslated this is because in Sotho it was decided not to translate the names of computer languages, they're really like brand names. So here the accelerator appears in English associated with Perl. Perl appears in the translation and therefore we could simply have transferred it without any worry about accelerator conflicts.
An ~example	Bad~translation	Good ~translation	This example shows a common error of leaving out a space between words. Because the accelerator sometimes in your mind looks like a space its easy to forget to place the space between words in the translation.

Checking

The pofilter tool has an accelerator test. This will check for missing accelerators as well as accelerators that shouldn't be in the translation. The tool can tell the difference between the various accelerator keys used.

```
pofilter --mozilla -t accelerators <original> <accelerator-errors>
```

This will check for Mozilla PO style accelerators (&) in the 'original' directory and output any errors to a new directory called 'accelerator-errors'

Please check the pofilter documentation for more details on how to use this tool.

Errors

What happens if you select the same accelerator key for two different components? How do you check this?

Firstly, it is not a big problem. Once the application is fully translated these conflicts should sort themselves out over time.

Application Specific Notes

KDE

FIXME There are settings that can be used in KDE to check for accelerator conflicts. Not sure if any testing is possible in Gnome, OpenOffice or Mozilla.

OpenOffice

OpenOffice.org seems to have a system that will automatically determine missing accelerator keys. What would work best with this system is to mark the accelerators that you would like to remain static and allow the others to be automatically determined. So keep "File", "Edit", "View" accelerators constant but allow all others to be determined at runtime.

1.11.6 Plurals

Believe it or not some languages have more than one plural form. This fact does come as a surprise to many, especially programmers.

There are two aspects to plurals that you need to be aware of:

- Those embedded in existing string eg. "file(s)"
- Properly defined plurals forms

Plurals embedded in strings

English adds **s** to the end of most words to form a plural. Thus you will see many instances where a programmer has simply add **(s)** to the word to indicate that it can be both singular and plural. Here is an example in Tsonga, in this case plurals add/change text at the beginning of the word, not the end. This often simply looks ugly.

```
"Show/Hide Axis Description(s)"
"Kombisa/Fihla (ti)nhlamuselo ya tikhona"
```

Furthermore, the grammar in other parts of the sentence might need to agree with either the singular or the plural case, but can't agree with both.

Here are the options available to you to deal with these type of plurals:

- If your language does not use plurals, you can just translate it disregarding the embedded plural.
- Use a similar construct if it works in your language.
- Abandon the singular and use only the plural form. This is what is done in Xhosa where a singular/plural form often involves prefixes so you end up with "(i)ifayili" or worse. So you simply use the plural "iifayili"
- Report the error to the programmer. Only use this option if this is the case for a valid plural i.e. the translator has used some kind of variable e.g. "%n file(s)" or two strings follow each other "%n file", "%n files"

Properly defined plural forms

Programs do have the ability handle plurals correctly. This is usually achieved by using the Gettext library or some similar method. When the application runs it will determine which plural form to use based on the number that is being displayed. So in English it would display:

- 0 files
- 1 file
- 2 files
- 3 files
- etc.

Gettext plurals

Gettext uses the "Plural-Forms" header in the PO file to define:

- `nplural` – the number of plural forms.
- `plural` – an expression which when evaluated determines which form is appropriate for that number. If a definition does not exist for your language then you will need to create one or get someone to help you.

Once this is defined then your PO editing tool will display the correct number of fields for you to enter the plural forms.

A list with these settings in some languages is available *here*. To find out how to define these entries, see Plural forms (gettext manual)

Historical KDE plurals

Note: KDE now uses standard Gettext plurals. This section is just for historical reference.

xxx

The plural form for KDE is defined in the kdelibs.po file. Choose one of the options or ask for help on their mailing list.

You will recognise KDE plural messages as they all start with "_`n: ``" and each form is separated by "n`"". in your translation you leave out the "_`n:  ``" and include as many translations as there are plural forms in your language, with each one separated by a "n`""

Language specific notes

- Arabic (in English)

1.11.7 Variables

Variables are placeholders that will be substituted with values when the translated application is run.

When you are translating and see a messages that looks like this:

```
Error: file %s causes error %s
```

You need to know that both of the %s will be replaced during execution to produce an error message such as this:

```
Error: file /home/bob/todo caused error segmentation fault
```

Variables styles

There are many different types of variables the most common are the printf style which include the following:

```
%s %d etc.
```

However, **Python named variables** are becoming more common:

```
%(name)s  %(filename)s  %(message)s  etc.
```

Do **not** translate the word(s) inside these named variables: these are literal expressions, not translatable terms.

Over and above these you will see various styles depending on the application that you are translating. Here are the styles present in the major applications:

OpenOffice.org

```
&file;
%PROGRAMNAME
%PROGRAMNAME%
$(file)
$file$
${file}
#file#
($file)
$[file]
[file]
$file
```

Mozilla

```
&name;
%name%
%s
$name
#1
```

GNOME, The Translation Project and others

```
%s
$(name)
%(name)s  [Python named variables]
```

What values will a variable contain?

Knowing that the %s will contain values will help you decide how to structure the sentence. In many variable styles the programmer will name the variable: file, name, dir. From this you can know what content to expect. With the case of printf style variables (those that start with % and have one letter) you can use the printf manpages to determine the type of variables but generally:

letter	content
s	string
d	decimal

Use this information to restructure your sentences.

Changing the order of variables

Your language might be one that has a different sentence structure or order to that of English. In this case feel free to change the order of the variables. However in your translation you will need to tell the computer what new order you have used. You can do that by numbering the variables:

```
Xxxxx %2$s xxx xxxx %1$s
```

This will reorder the variables placing the second %s before the first.

Please note that not all translated applications will support this feature. Almost all GNU Gettext based programs will allow you to do this. Others may not, so please check.

Leaving out a variable

It might be possible with some applications to leave out a variable entirely. Applications that use named variables might work. For glib based applications (applications using GNOME or GTK) you might be able to do the following (can somebody confirm?):

```
xxxxxxxxxxxx %0$s
```

This way gettext will still accept your translation as valid (it contains all the parameters) but the variable will not appear.

Languages that need to prefix variables

If your language adds prefixes to words depending on how they relate to other parts of the sentence then you need to choose how to add you prefix wisely. This should probably become a standard for your language. Here are some examples that give you an idea of how your translation would appear, the language is (almost) fictitious.

Language	Translation	When running	Comment
Default	Loading %s	Loading google.com	This is without any translation
1	Layisha %s	Layisha google.com	This works but is not what the language speaker expects
2	Layisha i%s	Layisha igoogle.com	This is correct according to the language but produces an unreadable result
3	Layisha i-%s	Layisha i-google.com	This balances what the language user expects while making it readable
4	Layisha i-%s	Layisha i-google.com	This might look a bit odd to the user but would also work

The main issue that you must consider is that you are often not aware of what content the variable might contain.

1.11.8 Translating Equations

These are not equations in the mathematical sense but here we refer to items in the translation that have an equals sign.

English	Form contains enctype=%S, but does not contain method=post. Submitting normally with method=GET and no enctype instead.
Afrikaans	Vorm bevat enctype=%S, maar bevat nie method=post nie. Dien soos gewoonlik in met method=GET en sonder enctype in plaas daarvan.

Usually these equations refer to real values that appear in a programs configuration file or they refer to items that would appear in the header of some protocol or email message. Therefore in the example above neither method=, enctype=. POST nor GET would be translated.

Lets say that again. The computer will understand method=POST but will know nothing about your translation "metode=POS". Also a user trying to get to the bottom of an email problem or to find a value in a configuration file will find method=POST they will not find "metode=POS".

However, it may be acceptable to offer a translation of the values to a user in brackets if you believe this will add clarity.

When should I translate the right hand side of an equation?

This will be on a case by case basis. In the example above you would not translate the right hand side (RHS) as it is a variable value. However if the following was presented.

```
Description=A program for highlighting widgets
```

then you would translate the RHS as is is clearly language specific text. You would not translate the LHS no matter how tempting.

KDE .desktop files

Another special case of translating the RHS are the translation of KDE .desktop files. These contain only equation like entries. They all start with "Word=" where word is usually one of "Name, Description,Keyword, etc". As above do not translate the LHS but do translate the RHS.

1.11.9 Program syntax and Spreadsheet functions

Often you will find examples of program languages (syntax) or spreadsheet functions in your translations.

Simple Policy

Need a simple policy on what to translate?

Never translate function names, spreadsheet functions or program syntax

What does this look like?

- Functions: not(), getLastAuthor(), sheet.Count()
- Spreadsheet Functions: SUM, AVERAGE, DDIST
- Syntax: FIXME examples? NULL
- SQL: INSERT, SELECT, WHERE, NOT

Why to translate or no translate

Program languages are in themselves their own language. So think of it as another language that is untranslatable. For instance when programming in Java using a Chinese programming tool you will still write in the English looking Java language.

The same applies to spreadsheets. However, in some countries they have had versions of software that are in deed translated. So the common functions, SUM, COUNT, etc are translated. It is probably best that you do not translate them unless they are in common usage. Remember that when someone uses your software in their languages and want to type SUM to add a column of figures and you have translate this, then they will need to know the translated function name. Could be very problematic if they already now the English version. So think of Spreadsheet functions as they same as a programming language.

1.11.10 Translating Example Paths and URLs

A path shows you where a file is located on your computer

```
/home/dwayne/somefile.doc
c:\Windows\Desktop\somefile.doc
\\Server\share\somefile.doc
```

A URL is similar but relates to files located on the Internet

```
http://www.google.com
ftp://someserver.com/file/to/download.zip
```

In translatable messages you will find real paths and URLs but you will also find instances where the author is giving the user an example.

Simple Policy

Want a simple policy on what to do with URLs (example or real)?

Do not translate ANY paths, URLs or URIs

System directories

Many operating systems contain system directories, these are directories used by the operating system itself. System directories will not change even when your language changes. The following are system directories on Linux and should not be translated:

```
/home
/mnt
/usr
/etc
/bin
/sbin
```

The following are system directories on Windows and should also not be translated:

```
C:\Windows
C:\Windows\System
```

Home directories

On Linux systems each user has a home directory which corresponds to their username and is usually located under /home eg. /home/fred. The word **home** should not be translated because it is a system directory. But the username should be translated. You should probably keep the username to ASCII characters (some operating systems cannot use Unicode in filename) and thus romanise the name if you needed.

Extensions

A file is made up of the:

fullname	/the/path/to/file.doc
path	/the/path/to/
filename	file.doc
basename	file
extension	doc

The file extension is thus the characters that appear after the last fullstop in a filename. You do not translate the file extension but you do translate the basename. Eg. budget.doc – Translate **budget** but do not translate **.doc**.

Examples

Often examples can be identified because they involve saving a file from within an application. Eg, "Save the picture to /home/dwayne/picture/snapshot.png" You will want to translate dwayne, picture and snapshot. You would not translate home because that is a system directory nor png because it is an extension.

URLs

A URL is made up of these components

```
protocol :// server : port / directorie(s) / filename . extension
```

Here is a real URL, i.e. one that you would actually use:

```
http://www.translate.org.za/downloads/openoffice.org-afrikaans.tar.gz
```

you should not translate it unless you needed to point to a version of the URL that is in your language.

This is an example URL, i.e. one used as an example to a user:

```
http://example.com/directory/filename.html
```

you would translate: example, directory and filename.

You would **not** translate:

http	the protocols are not translatable
.com	top level doamns are also not translatable
.html	is a file extension and therefore not translated

1.11.11 Translating HTML

HTML is used to design web pages, but it often occurs in GUI translations. You will also see XML – which looks very much like HTML. They are both examples of markup languages used to specify some information about text. This guide explains which parts of HTML you can safely translate and which you should leave unchanged.

HTML is recognised by tags between angle brackets: <tag>

A tag can also contain extra information: <img src="picture.jpg" /> Here the img (image) tag contains an extra attribute to specify the file name of the picture to show in the image tag.

Tags often occur in pairs: <p> </p> – note the difference between the opening and closing tag.

What not to translate

Do not translate the actual markers. HTML consists of tags which indicate the start and end of a section of text. This text could be a heading, a paragraph, a hyperlink or just a piece of text to display in bold:

```
<h1>A heading</h1>
<p>A paragraph</p>
<a href=bob.html>A hyperlink</a>
This is normal and <b>this is bold</b>
```

Some markers just beg to be translated, such as these:

```
<title>, <center>, <body>
```

Do not be tempted – these need to remain in English.

Reordering or changing tags

In some cases, it might be necessary to reorder the tags. If tags surround some text, it means that the translation representing the text inside the tags should probably also be surrounded with the same tags.

Some inline tags can represent something that should be positioned in the translation, such as an image or a footnote. Take care where you put this in the translation to have the same effect as the original text.

Whenever you work with tags, always ensure that opening and closing tags go together in pairs, and that the closing tags always follow the opening tags.

Attributes – which to translate

An attribute is a variable associated with a tag. E.g. <body bgcolor=blue>, here "bgcolor" is an attribute and "blue" is its value. Attributes, like tags, are never translated. However some values can be translated. In the example above the value "blue" should not be translated.

There are only a few values that can be translated:

alt	found in the img (image) tag and used to give a textual description of the picture that will be loaded. This is essential for people with disabilities
title	a text title that pops up when you hover something, exactly like a tooltip

In some cases you might also need to change the "lang" or "dir" attributes. This is best left to people with good knowledge of HTML.

Tags that should be translated

Some things that looks like tags are not really tags and should be translated. These would include the following: <Error>, <File not found>, etc.

How to identify them? If you are an experienced HTML editor you will immediately know which are real tags. If you are not, then use this rule: If its all in lowercase or uppercase then it very likely a tag. If it combines case most likely it is not. If it contains any attribute such as <font color=blue> it is definitely a tag. And lastly if it's the only text in the message e.g. "<None>" then it is most probably not a tag.

Should I change hyperlinks and images?

Although as a translator you should only change the text of the program, you do have complete control over the HTML. This can be used to your advantage if needed. If for example you were translating a manual that referred to an image of the application then you would most likely want to have an image in the users language, rather then explaining to them in their language using an English picture.

Because of the work involved, the translation teams will often not change images until much later in their efforts.

To change the image first you will need to create an image. You can use ksnapshot or gnome-screenshot or Gimp to create screenshots. Remember to keep the look consistent, i.e. try to use the same theme across all images. Ideally your picture should be the same size as the one it is replacing.

Lastly depending on the application you will need to place the file in the correct place so that it will be shown with your translations. What usually happens is that images are first sought in the language specific directory and if they are not found then they use the English version.

1.11.12 Escaping

Escaping is a method that allows us to tell a computer to do something special with the text we supply or to ignore the special function of a character. To tell the computer that it should expect an escape character we use a special escape marker, we usually make use of the backslash (). So an escaped sequence consists of the escape marker followed by another character.

Although many of these are hidden from you in some tools (such as Virtaal), you might still encounter them from time to time.

In most cases you will see from the original source text when these should be used, and in many cases you can follow the example of the source text closely.

Types of Escape Characters

Each escape sequence has a special meaning:

Escape Sequence	Effect	Description
`\n`	newline	Adds a newline at that place in the text (similar to pressing `Enter`). It might be shown with the ¶ in your translation program.
`\t`	tab	Add a tab marker at that spot in the text. This is sometimes used to align text in columns.
`\r`	carriage return	Return without advancing the line This is usually used with n in software for Windows.
`\\`	Print a real backslash	If you need a real backslash then you need to escape the backslash. The computer will only print one backslash.
`\"`	double quote (")	In some cases a double quote has a special meaning, and should be escaped if you really want a double quote.
`\uxxxx`	Unicode character	Sometime you want to refer to Unicode characters using normal ASCII. Many programs and programmers use u to refer to the Unicode character. Where xxxx is the hexadecimal or decimal value for the specific Unicode character.
`&apos;`	apostrophe (')	In some XML documents, you might need to represent the apostrophe and some other characters with XML entities. For example '<a href="index.html" title="Rock &apos;n Roll">link</a>'

Escape Characters in Your Translations

What you see	What it means	What you do
…server active.\nDo you want…	This is easy to recognise as the newline appears between two sentences. When the program runs the sentences will appear on different lines.	The best option is to place your n escape in the same place
…jobs and/or\nOLE actions…	This is harder to recognise. There will be a line break between "and/or" and "OLE".	Look at the structure of the whole translation, in a case like this the author is using newlines to balance the text (i.e. make the lines appear to be equal in length). If that is the case then balance your translations in a similar way.

Dos and Don'ts

- DO
  - Copy the escape exactly as you see it.
  - Move the placement of escapes if they are being used to balance sentences
  - Make sure you have exactly the same number of escapes as in the original
  - Drop " or ' escapes if your language does not use the "double" or 'single' quote method of emphasis that English uses. Be aware that the quality checks by some tools might complain, although the *doublequote* and *singlequote* tests of pofilter should be aware of your language's written rules. Let us know to try to take your language's rules into account.

  - Place " or ' around the same **concept** as the English i.e. around the same word or words that the English had surrounded.

- DON'T
  - Change the escape in any way. Do not use **/n** instead of **n** just because you can't find **** on your keyboard
  - Confuse sentences like this "File Open New" as an escape. If they are not in the above list then they are most likely not escape characters.
  - Add or delete spaces around the escape. If it's there, keep it. If not, don't add it.

1.11.13 Casual Translations

Computers are rather formal. In fact if they were too friendly we wouldn't enjoy using them. You don't expect a computer to talk to you as a friend you expect it to give you the information you need with the minimum of fuss and then get out of the way:

We prefer: The file could not be found

To: That file that you were looking for I can't seem to find it. Help!

In your translations you have the opportunity of removing language that is too casual. Be careful when translating games and kiddies programs as you might want to keep informal language.

Here is a snippet of an email from Malcolm Hunter – one of the British English translator. You might want to take note of his points and remove these from your translations.

```
Something I've forgotten to mention:

There was discussion sometime ago about use of first person and casual/
IRC-style usage. As much as possible can we try to make the wording more
professional.

1. I, me, my should be avoided, unless it is the user's response (eg. "Let me
choose") and can't easily be changed to something else. The program shouldn't
be saying to the user: "I was unable to...", etc.

2. We is another one, there are 2 examples of this in kget.po ("We are
online / offline") - who's "We"? I'm not sure what to do with these at the
moment - depends on the context.

3. The program should not apologise - there is a Sorry function, but the
actual text shouldn't have the word "Sorry" in it. I'm not sure about
"Please" either.

4. No IRC emoticons, abbreviations (eg. LOL) - we might get developers
moaning, but at the end of the day these apps are being used by corporate
users as well.

5. Same goes for casual, conversational wording (eg. "Oops!" which I recently
found in a dialog title bar). Same reason as item 4.

With games and amusements where the target user-base is mainly home, we can be
a bit more relaxed with this as we don't want to spoil the fun aspect.

Thanks,
Malcolm
```

1.11.14 Capitalisation

What should you do about capitals that you see in the English source translations? You will notice that English makes use of *Title Case*:

> Save As… Find Next Bookmark This Page…

Note how almost every word starts with a capital letter. This is a style often used in English software, especially for titles, buttons and menus. This is also sometimes called "header capitalisation"

What do you do in your language?

Choosing your style

The first thing to remember is that you should be adapting the software to the conventions of your culture and language. You do not need to slavishly follow English title case. By all means if your language follows this convention then use it. But if is doesn't, then please use the normal case style used in your language. Very few languages actually use the style of title case as much as English do. Rather just follow the normal rules for capitalisation in your language.

Starting capitals or the lack thereof

You will notice that certain sentences do not start with a capital. Try to match these capitals as they are often done for a reason. Sometimes they represent phrases that will be joined at runtime (this is a bad practice and should be reported to the programmer for correction).

To check your starting capitals use pofilter

```
pofilter -t startcaps input output
```

What to do with 'OK'

In English the **''OK''** or affirmative button is all in capitals. It is thus tempting to provide a fully capitalised translation in Xhosa that woul dbe **''KULUNGILE''**. But this looks strange. Rather abandon the English capitals and use normal capitalisation for your translation ie **''Kulungile''**

1.11.15 Selecting and Creating new words

Your language does not have a word for a certain computer term – what do you do? Many people decide to leave the term in English. However, please take a moment to think about why you are translating this software. Will people who use your language be able to identify or understand the English word? Will they even be able to read it? If the answer is no then you need to look at creating or adapting the word.

Choosing a new word

- **Find the correct translated word.** This is the simplest and best approach but often however this comes overs time. You may need to go back and revise word choices to eliminate confusion. For instance the same word for Cancel and Delete was used in our Xhosa translation until the translators grasped the potential confusion and corrected the mistake. If you didin't understand the implications in this example consider that 'Cancel' is a safe option (go back and ignore everything I was doing) while 'Delete' is a destructive option (you will remove files from your computer). Clearly you need very different words.

- No word exists. **Think about what the word means.** Make use of dict.org, dictionary.com to check on the word's definition. Be careful with some words eg. 'Port' which have very different meanings in the computer world. Once you understand the meaning of the words and the idea that needs to be conveyed try to identify a phrase or word that conveys the same meaning in your language.
- Nope nothing. OK you're stuck with an English word which you can't or don't want to translate. **Try transliteration.** That is take the word and change its spelling and pronunciation to better match your language. This way at least people familiar with the language will be able to read and pronounce the word. English language speakers have been doing this for a very long time.
- Still nothing. **Leave it in English.** But remember who your translating for and the problems this may cause them.

Acronyms

Acronyms are words formed from the initial letters of phrases.

- HTTP – HyperText Transfer Protocol

We often choose to leave these as they are as the meaning of the acronym is to most people disasociated with the actual words that make up the acronym. Some languages do choose to define

the expanded acronyms in their language but keep the original English acronym. Finally some languages have translations of these acronyms, use them if you feel that they are widely used and understood by your audience. Languages that fall into this category usually have well established and government supported terminology development units or language boards

You need to weigh up whether a translated acronuym will help or hinder. Not being aware of http may cause some people to battle with accessing the internet.

Notes

The following email reply from Chusslove Illich (caslav ilic at gmx net) gives some more pointers when creating new words.

```
> [: Dwayne Bailey :]
> But don't be too paranoid as even if they're not the best word they
> are merely place holders for ideas.  People will learn what these
> ideas are.

I think this is very important point. Often, translators try to think of
translation which is "intuitively understood", "friendly to the
beginners", etc. In my opinion, these considerations are almost
counter-productive, as then one easily ends up with translations which are
of questionable usability -- not determining original idea uniquely enough
or simply too long.

An example of this from my language would be "formal translation" (ie.
currently used in translation of Windows, KDE and Gnome) of "driver",
which if translated back to English is "management program". It is both
ambiguous (eg. "printer management program" is most certainly not "printer
driver") and way too long.

I think it is better to, whenever possible, draw ideas and associations
from the fields you know to *work*: English language itself, other
languages with long tradition in science, older branches of science in own
native language.
```

(continues on next page)

(continued from previous page)

```
Again, in the example of "driver" in my language, there actually exists an
excellent (IMO, of course) translation, and that is the literal one (of
"driver" in "cattle driver", not in "bus driver"); it is very short (2
syllables, 5 letters only), already used in my language in eg. mechanical
engineering, used in many European languages (eg. German "der Treiber"),
and keeps the same association as in English (it "drives" the device).

> You might want to involve some linguists in the translation as they
> will be better equipped to select and adapt words.

This would also be a great advantage. Whenever you want to use
transcription of English word, or coin a new native word, linguist can
tell you whether it fits well into spelling and pronunciation rules of
your language.

Also, he can signal whether a coined word is following the language
tradition (ie. will not sound too unnatural to the speakers) or not. Eg. in
my language it is not traditional to make double-noun or noun-verb
composite words, but there is a large set of usual prefixoids/suffixoids
that can be used.
```

1.11.16 Translating Brand Names

By brand names we mean Trademarks but also some other recognised but not, yet, trademarked names. Brand names are emerging as the differentiator within Free Software and people are becoming protective over them, i.e. they don't like you translating them or using them without permission. Examples include (registered or otherwise) Mozilla, OpenOffice.org, KDE, Red Hat, Debian, etc.

Most brand names are easily left as is while some are tempting to translate:

- **OpenOffice.org**: Writer, Calc, Draw, Impress – are all translatable. You team needs to define its policy on this..
- **Mozilla**: Editor, Mail & News – are all translatable. Unlike The OpenOffice.org case these are probably best translated. A user would say I edit my webpages with the Mozilla Editor. Unlike OpenOffice.org were a user might say 'to do that you need to open Calc and go to…'. Editor is much more generic than the OpenOffice.org examples.
- **KDE**: kaddressbook – an unoriginal name for an application. This probably should be translated. But be aware a user can only run kaddressbook from the command line by actually typing 'kaddressbook' it will not work if they type the translated name, this is not an issue if your users make use of the GUI and not the command line.

Simple Policy

Want a quick and easy policy on brand names?

Do not translate any brand or application name

Policy ideas

- If it works in your language leave it
- If it is an obscure application maybe change it

- Think of derivatives. Eg. OpenOffice.org has Draw which you might want to keep and Drawing for drawings produced by Draw which you would want to translate. Does this make it confusing to the user?
- Be consistent. Writing down your policy and reasons saves you having to explain things a hundred times. And policies can be changed.
- If needed, think about transliterating the name so that people can understand how it is pronounced. This is similar to the idea of transliterating programmers' names.

1.11.17 Program Names

Your language team needs to decide if you are going to translate program names. There are a few approaches you can follow to make untranslated program names accessible to non-English speakers.

Simple Policy

Need a simple policy on program names?

Do not translate program names

Pros and Cons of Translating Program Names

Pros	The program is shown as a script/pronunciation that the user understands
Cons	A user will not find the translated program anywhere on the system. I.e. they will not be able to run the program using its translated name

Some tips

- **Use single quotes**. It is quite easy to use single quotes around the program to indicate that it has not been translated e.g. "Click on the 'KFishMonger' icon"
- **Change sentence structure**. This source string "%s: Can't init pipe to gdmgreeter" can quite easily be translated as "%s: Can't init pipe to program gdmgreeter", note that by adding the word program it adds clarity
- **Use explanatory brackets**. If needed add explanatory brackets after the word. You have two options just stick to one method throughout:
  1. Translated_Program (English_Original)
  2. English_Original (Translate/Transliteration).

1.11.18 Punctuation

Punctuation is used to indicate various things in program menus e.g.:

- Ellipses (. . .) – a dialogue box will appear for further input e.g. "Save As. . . "
- Colon (:) – usually in a list of entry fields e.g. "Name: "

And sometimes it is simply there for style.

Simple Policy

Want a simple answer to punctuation?

If your language uses similar punctuation symbols to English, **follow the English punctuation exactly**

Specific Punctuation Issues

Ellipses (. . .)

Ellipses are usually used to indicate that a dialogue box will appear and the user will be required to give further input. Compare the menu entries **Save** with **Save As. . .** . You’ll notice that ‘Save’ has no ellipse, the computer will immediately save the current file and needs no further input from the user. While ‘Save As. . . ’ has an ellipse indicating that the user will be presented with a dialogue box, this time the Save dialogue box so that they can supply a new filename.

Make sure that you include all ellipses in your translation. You might find that a programmer has used 2 or 4 dots instead of 3: use 3 dots in your translation and report the error to the programmers.

.. –> . . .
. . . . –> . . .

The ellipse always occurs at the end of a message to indicate continuation. Even if you need to change the word order for your language make sure that the ellipses occurs at the end of the sentence.

Blah blee. . . –> Flee. . . flah is **WRONG**
Blah blee. . . –> Flee flah. . . is **CORRECT**

Fullstops (.)

We all learnt that sentences should always end in a fullstop. However, when translating add a fullstop only if one appears in the English. If you do not see a fullstop then do not add one. The reason for this is simply style, there is a most probably a reason why the fullstops have been left out, it looks better. If during testing you notice that it doesn’t work or that there should be a fullstop then correct appropriately and report the bug. But in the first translations simply follow the English.

Colons (:)

Colons are usually followed by a space and are used to indicate that this is a label for the adjacent text input box. Please make sure that you include the colon and the space is required.

French Style

It seems to be tradition that the French and Vietnamese place a space between a word and :;!# e.g.:

```
You will delete all your files !
Name :
```

French also uses a space before ? – a question mark. Most languages do not follow this style, so be sure to use the accepted punctuation style in your language.

One exception is when the last word is a variable:

```
Are you sure you want to access %S ?
```

Which might display as:

```
Are you sure you want to access http://example.com ?
```

In this case it is good that the question mark is separate as it ensures that you do not think that the question mark is part of the URL or file name filled into the variable during runtime.

Full-width punctuation

Some languages, notably Chinese and Japanese, sometimes use full-width punctuation marks instead of the ones normally used for other languages. It is also common to **not** use spaces after such symbols, since the full-width punctuation is usually displayed to be properly spaced. The following table shows some possible examples where the punctuation is changed, and the space is removed after the punctuation mark. The way it displays could be influenced by the fonts used to display this.

English	Full width
word. word	
word; word	
word: word	
word! word	
word? word	

Checking

You can check your start and end punctuation using pofilter

```
pofilter -t startpunc -t endpunc <input> <output>
```

Other useful tests are:

```
brackets
startwhitespace
endwhitespace
```

1.11.19 Problem Words

These are some words that always create problems in that they are often translated into a language using the same word but these words have different meanings in computer terminology.

How to check for duplicates

You can use pocompendium -v to see if you have used the same word for various English words.

In and Out of your computer and programs

The following are in order of severity

Word	Meaning
Off	Switch an option off (like switching off a light switch)
Close	Usually closing the Window of an application but not closing the application itself (close a window)
Quit/Exit	Close the application (exit from the stage)
Logout	Close all the applications and return to the login screen (go outside)
Shutdown	Shutdown all systems on the computer and often turn it off (turn the light off)

Getting rid of things

Word	Meaning
Cancel	Ignore this action
Delete	Remove a file
Stop	Do not continue with this task
Erase	Same as delete
Remove	Same as delete
Clear	Usually to clear the screen, to erase a white/black board

Your settings, etc

Most of these are the same, although some programs might use them slightly differently.

```
Preferences
Settings
Configuration
Options
```

Programming languages

You will often see text that relates to programming or programming like languages. Some of them you can change.

- Spreadsheet functions – you can translate these if they make sense
- Program syntax – never translate as this is the grammar of a programming language
- SQL – you should never change this unless the program is providing some user interface that will hide people from the actual SQL language.

Do translate

- Many people cannot read Latin characters so the words and names will appear meaningless
- You don't need to think about whether this is or isn't translatable

Don't translate

- Think of it as another language, you have to learn the language of the spreadsheet
- Outside of your application people will see only the English so if you translate it will confuse them
- If people work in a multilingual environment its better to leave it untranslated becauase you don't want them to struggle when they can't remember the Afrikaans naam for SUM()

1.11.20 British English

Most programs are written using American English spelling. It is a simple process to change the spelling to British English to satisfy the eye of English speakers in Canada, Australia, South Africa and other commonwealth English speaking countries.

References

These are useful references to the difference between the various English spellings.

- http://www3.telus.net/linguisticsissues/BritishCanadianAmerican.htm
- http://www.musicalenglishlessons.com/spelling-diffs.htm

Tools

To initialise all the POT files to be fully translated:

```
for pot in `cd templates; find . -name "\*.pot"`
do
  mkdir -p en_GB/`dirname $pot`
  msginit --locale=en_GB --no-translator -i templates/$pot -o en_GB/`dirname $pot`/
↪`basename $pot .pot`.po
done
```

To examine all potential American spellings use pofilter:

```
pofilter -t musttranslatewords --musttranslatefile=en_US-words-to-change en_GB en_GB-
↪check
```

If a msgid contains any of the words listed in the files *en_US-words-to-change* then these will be extracted to en_GB-check. Correct them and then run:

```
pomerge -i en_GB-check -o en_GB -t en_GB
```

CAUTION

This will not check for bad grammar, sentence structure, humour, etc. You need a person to do that. But its a start.

1.11.21 Testing Translations

How do you test the translations that you have created? In some cases it is almost impossible to quickly test since you need to compile the application. The following are guides for various systems to allow you to quickly test your translations.

Cannot test

These cannot be quickly tested:

- OpenOffice.org – requires the application be recompiled
- Mozilla Firefox and Thunderbird – requires the creation and installation of an XPI

Gettext

This works on Linux or any system where there is a default location for the compiled PO files.

```
$ mkdir -p $HOME/.local/share/locale/$your_language_code/LC_MESSAGES
$ pocompile $your_application.po $HOME/.local/share/locale/$your_language_code/LC_
↪MESSAGES/$your_application.mo
```

Substitute $your_language_code for your iso639 language code e.g. zu for Zulu. And substitute $your_application for the application that you are translating (actually this is the name of the text domain which might be slightly different from the application name but is usually the same as the PO filename).

Now we have a local repository for storing our custom .mo file. And we have compiled one and placed it in the correct location. You can substitute `pocompile` with `msgfmt` in the above example.

```
$ export LANGUAGE=../../../$HOME/.local/share/locale/$your_language_code:$your_
↪language_code
$ $your_application
```

This will firstly force us to look into our custom location for MO files and use those first then fallback to those in the system location. Now when you run the application you will see your translations and can quickly test them.

- Per language *Translation Guidelines*
- Online resource for *Defining Words*
- When translating
  - *Common translation errors* with *Common Translation Errors*
  - *Accelerator keys*
  - *Plurals*
  - *Variables*
  - *Equations*
  - *Program syntax and spreadsheet functions*
  - *Paths and example URLs*
  - *What bits of HTML to translate*
  - *Escaping*
  - *Casual Language*
  - *Capitalisation*
  - *Creating new words*
  - *Brand names*
  - *Program names*
  - *Punctuation*

  - *Words that you should watch out for*
- *British English*
- *Testing Translations*

1.12 Locales

1.12.1 About locales

Locales are definition files that tell a computer about the cultural conventions of a language and/or country. So a locale will define how characters are sorted, how the date and time are represented, the names of the days of the week and months of the year.

These are the locale files that interest a localiser:

- glibc
- ICU
- OpenOffice.org
- CLDR

To help with defining locale data you will also need to know about:

- Character sort orders (collation)

The CLDR is a unification of the various locale information that uses the locale data markup language specification. The CLDR is an important project to contribute your locale data to as this will eventually become the authoritative repository for locale information.

1.12.2 Locale Resources

- Common Locale Data Repository
- Alphabets (what characters appear in a languages alphabet)

The following are useful general resources that can be used during the construction of your locale files:

- Month names in various languages
- Day of week names in various languages
- Language names in their original form
- Country details names (replace with your country ISO code in the URL)
- Car license plate numbers
- Dialing codes
- International call prefixes (ie what you use to dialout)
- ISBN book numbers (2)
- ISO 3166 2 letter country code and 3 letter codes (not many really):
  - ISO 639 language codes (Request a new code)

Collation

- Language collation or character sort orders
- Some other alphabets

Language Specific

- Venda:
  - Notes on unicode characters
  - Font that can be used for Venda chars
    * http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&item_id=Gentium_linux

Operating System Specific

Microsoft Windows

Microsoft Language IDs

This is a list of languages and their identifiers as assigned by Microsoft and used in Microsoft products.

Language	Primary ID	Sub ID	Language Code
Afrikaans	54	1	afk
Albanian	28	1	sqi
Arabic (Saudi Arabia)	1	1	ara
Arabic (Iraq)	1	2	ari
Arabic (Egypt)	1	3	are
Arabic (Libya)	1	4	arl
Arabic (Algeria)	1	5	arg
Arabic (Morocco)	1	6	arm
Arabic (Tunisia)	1	7	art
Arabic (Oman)	1	8	aro
Arabic (Yemen)	1	9	ary
Arabic (Syria)	1	10	ars
Arabic (Jordan)	1	11	arj
Arabic (Lebanon)	1	12	arb
Arabic (Kuwait)	1	13	ark
Arabic (U.A.E.)	1	14	aru
Arabic (Bahrain)	1	15	arh
Arabic (Qatar)	1	16	arq
Armenian	43	1	hye
Assamese	77	1	asm
Azeri (Latin)	44	1	aze
Azeri (Cyrillic)	44	2	azb
Basque	45	1	euq
Belorussian	35	1	bel
Bengali	69	1	ben
Bulgarian	2	1	bgr
Catalan	3	1	cat

Continued on next page

Table 1 – continued from previous page

Language	Primary ID	Sub ID	Language Code
Chinese (Taiwan)	4	1	cht
Chinese (P.R.C)	4	2	chs
Chinese (Hong Kong)	4	3	chh
Chinese (Singapore)	4	4	chi
Chinese (Macao)	4	5	chm
Czech	5	1	csy
Danish	6	1	dan
Dutch (Netherlands)	19	1	nld
Dutch (Belgium)	19	2	nlb
English (USA)	9	1	enu
English (GB)	9	2	eng
English (Australia)	9	3	ena
English (Canada)	9	4	enc
English (New Zealand)	9	5	enz
English (Ireland)	9	6	eni
English (South Africa)	9	7	ens
English (Jamaica)	9	8	enj
English (Caribbean)	9	9	enb
English (Belize)	9	10	enl
English (Trinidad)	9	11	ent
English (Zimbabwe)	9	12	enw
English (Philippines)	9	13	enp
Estonian	37	1	eti
Faeroese (Faeroe Islands)	56	1	fos
Farsi	41	1	far
Finnish	11	1	fin
French	12	1	fra
French (Belgium)	12	2	frb
French (Canada)	12	3	frc
French (Switzerland)	12	4	frs
French (Luxembourg)	12	5	frl
French (Monaco)	12	6	frm
Georgian	55	1	kat
German	7	1	deu
German (Switzerland)	7	2	des
German (Austria)	7	3	dea
German (Luxembourg)	7	4	del
German (Liechtenstein)	7	5	dec
Greek	8	1	ell
Gujarati	71	1	guj
Hebrew	13	1	hbr
Hindi	57	1	hin
Hungarian	14	1	hun
Icelandic	15	1	isl
Indonesian	33	1	ind
Italian	16	1	ita
Italian (Switzerland)	16	2	its
Japanese	17	1	jpn
Kannada	75	1	kan

Continued on next page

Table 1 – continued from previous page

Language	Primary ID	Sub ID	Language Code
Kashmiri (Indien)	96	2	(6002)
Kazak	63	1	kaz
Konkani	87	1	kok
Korean	18	1	kor
Latvian	38	1	lvi
Lithuanian	39	1	lth
Lithuanian (old)	39	2	ltc
Macedonian	47	1	mki
Malay (Malaysia)	62	1	msl
Malay (Brunei/Darussalam)	62	2	msb
Malayalam	76	1	mal
Manipuri	88	1	(58)
Marathi	78	1	mar
Nepalese (Indien)	97	2	(6102)
Norwegian (Bokmal)	20	1	nor
Norwegian (Nynorsk)	20	2	non
Oriya	72	1	ori
Polish	21	1	plk
Portuguese	22	2	ptg
Portuguese (Brazil)	22	1	ptb
Punjabi	70	1	pan
Romanian	24	1	rom
Russian	25	1	rus
Sanskrit	79	1	san
Serbo-Croatian (Latin)	26	2	srl
Serbo-Croatian (Cyrillic)	26	3	srb
Sindhi	89	1	(59)
Slovak	27	1	sky
Slovenian	36	1	slv
Spanish (Castilian)	10	1	esp
Spanish (Mexico)	10	2	esm
Spanish (Modern)	10	3	esn
Spanish (Guatemala)	10	4	esg
Spanish (Costa Rica)	10	5	esc
Spanish (Panama)	10	6	esa
Spanish (Dom. Republic)	10	7	esd
Spanish (Venezuela)	10	8	esv
Spanish (Columbia)	10	9	eso
Spanish (Peru)	10	10	esr
Spanish (Argentina)	10	11	ess
Spanish (Ecuador)	10	12	esf

Continued on next page

Table 1 – continued from previous page

Language	Primary ID	Sub ID	Language Code
Spanish (Chile)	10	13	esl
Spanish (Uruguay)	10	14	esy
Spanish (Paraguay)	10	15	esz
Spanish (Bolivia)	10	16	esb
Spanish (El Salvador)	10	17	ese
Spanish (Honduras)	10	18	esh
Spanish (Nicaragua)	10	19	esi
Spanish (Puerto Rico)	10	20	esu
Swahili	65	1	swh
Swedish	29	1	sve
Swedish (Finland)	29	2	svf
Tamil	73	1	tam
Tatar	68	1	tat
Telugu	74	1	tel
Thai	30	1	tha
Turkish	31	1	trk
Ukrainian	34	1	ukr
Urdu (Pakistan)	32	1	urp
Urdu (India)	32	2	uri
Uzbek (Latin)	67	1	uzb
Uzbek (Cyrillic)	67	2	uzc
Vietnamese	42	1	vit

- *Language IDs*
- Locale IDs (LCIDs)
- Software to make a custom locale for Windows Vista

1.12.3 Glibc locale files

Locale files define the culural conventions of a language and region. The Glibc locale files are used on all GNU/Linux systems so are needed for software that will run on Linux.

Data

When creating loclale data for South African locales the data files are stored in version control. While these files are in development the reference version is the one contained there. When the files are suitably mature they are submitted to glibc and those become the reference versions while those located in version control will then contain any development work if needed.

In the mature stage the locale data can be abandoned and development made against versions in glibc.

It is suggested that you follow a similar model.

Tools

The following are tools located in the the Translate.org.za Subversion repository. They are useful for building and testing glibc locales:

missing	determines if a locale file has a certain locale field or not.
error	displays any compilation errors detected
install	performs a test install (use -r for a real install – must be root)
definition	prints the value of a locale field (installed locales only)
locale-escape	converts your locale into <UNNNN> format
check-dates	prints a list of the LC_TIME defined date formats for the locale

Editing

If you edit your locale file using vim then you can make use of the fdcc file highlighter. Newer versions of vim should already have this file installed and will detect the filetype automatically. If your file is not automatically highlighted then you will need to download the file and follow these instructions

It is preferable to edit your locale in UTF-8 and then use locale-escape to encode your work in the <UNNNN> format used in glibc locale files. Use locale-escape as follows:

```
cat unescaped-locale | ./locale-escape > escaped-locale
```

You can also use the iconv tool to achieve the same escaping (this will only work if your version of iconv supports the –unicode-subst option):

```
$ echo ÐÑÑÑÐÐÐ |
  iconv -f UTF-8 -t ASCII --unicode-subst='<U%04X>'

<U0420><U0443><U0441><U0441><U043A><U0438><U0439>
```

% Charset

All locales should contain a comment like

```
% Charset: UTF-8
```

at the beginning.

The recommendation is not to use ISO-8859-1 as it's outdated, unless there is a *substantial* install base. Ideally you should only use UTF-8. But if necessary, ISO-8859-15 is preferred above ISO-8859-1.

Checking

For a quick check first install the locale as root run:

```
install -r xx_XX
```

Then run checks, either

```
definition xx_XX
```

Or

```
definition -c LC_TIME xx_XX
```

And go through each one checking that the entries are correct

Defining LC_TIME

Use 'man date' to see what variables are valid in a locale file date and time formatting. If you want to remove space padding then use minus in the variable eg: %-e will print the day of the week without a space padding before the number. E.g. '[space]1' becomes simply '1'

Gentoo users

On gentoo you should NOT use the install script, but rather execute

```
localedef -i <your_escaped_file> -f UTF-8 <your_locale>.UTF-8 -c -v
```

The install script uses locale-gen and can give you quite a lot of trouble with accented chars.

Resources

Website and such:

- glibc locales mailing list
- glibc website
- glibc bugzilla

Producing you locale file:

- A site dedicated to glibc locale files and their creation
- A guide developed by the Lugandan locale creator
- The latest Afrikaans and Zulu locales both heavily commented with references for certain locale data

Resource specific to the ISO standard for locales files:

- The ISO 14652 standard that defines locale or fdcc files (seems to be the latest version of the standard)
- Comments from Ulrich Drepper the creator of locale implementation in glibc
- Keld Simonsen's comments on Drepper's mail (Simonsen manages the ISO 14652 standard)

Notes

All changes to glibc locales must also be reflected into the IBM ICU locales. So you need to post 'bug' reports against ICU and possibly against the OO locales as well.

Submitting your new/update locale to glibc

Note: double check everything before sending. Its easy to overlook silly things like comments that still apply to a previous language. Check them all again.

Officially you should send your locale files to:

- http://www.gnu.org/software/libc/bugs.html or
- bug-glibc@gnu.org or
- Use the glibcbug script which seems to email glibc-bug-reports-stable@gnu.org

I have in the past sent email to Ulrich Drepper, the glibc maintainer. This is not guaranteed to work but if all else fails try this route.

Attach the file and preferably a diff between your update and the one in glibc CVS

```
diff -u xx_XX.glibc_version xx_XX.updated > xx_XX.diff
```

Make the subject very clear: "Update xx_XX glibc locale file". Attach the files and send.

You also need to patch against localedata/SUPPORTED so that you can define what charsets you can use with your locales.

1.12.4 KDE Locales

KDE, because it runs on platforms other than Linux, has created its own locale information environment. The data stored is similar to other locales, including date and time formats, etc.

Resource

- README for KDE locale files and all the fields in the entry.desktop file
- Direcory of all current KDE country locales
- South African locale as an example

entry.desktop file

Most of the entries are self explanatory. All of the initial **Name[xx]** entries are supplied by translators for those languages and are most probably already complete.

Note that **Languages** defines what languages are used in this locale. Make sure that the first entry is the language that you want to be the default for your country.

1.12.5 Locales on X11

X11 it seems does not define its own locales but makes use of the system locale. It may only work for a subset of your systems locales because it needs to know about your locale. You need to make a few additions to ensure that X11 will recognise your glibc locales.

Various Servers

There are currently two Xservers for the Free Software world: XFree86 and X.org. XFree86 is rapidly falling out of favour but in order to cover all of the Linux distributions you will probably want to patch both of these.

Where to find the files to change

You can view these files using XFree86 CVS web and X.org CVS web

We have used the Kinyarawanda example here to illustrate what needs to be changed. You can read the bugzilla report for adding Kinyarawanda to XFree86. Here is another example for X.org

What needs changing

locale.alias

The file locale.alias is usually found here:

```
/usr/X11R6/lib/X11/locale
```

This needs to be updated to add all the locale permutations. Look at a language similar to yours and add the various permutations. Here are the lines that were added to get Kinyarawanda working:

```
rw_RW:                  rw_RW.ISO8859-1
rw_RW.iso8859-1:        rw_RW.ISO8859-1
rw_RW.ISO88591:         rw_RW.ISO8859-1
rw_RW.ISO-8859-1:       rw_RW.ISO8859-1
rw_RW.utf8:             rw_RW.UTF-8
```

If you are not sure what character sets will be valid then look here to determine what will work:

- http://www.inference.phy.cam.ac.uk/dasher/download/alphabets/ALPHABETS.html
- http://www.eki.ee/letter/

compose.dir

```
iso8859-1/Compose:      rw_RW.ISO8859-1
```

and also

```
en_US.UTF-8/Compose:                    rw_RW.UTF-8
```

Although it looks like the UTF-8 ones do not currently work

locale.dir

```
iso8859-1/XLC_LOCALE:                   rw_RW.ISO8859-1
```

and also

```
en_US.UTF-8/XLC_LOCALE:                 rw_RW.UTF-8
```

xc/programs/Xserver/XpConfig/Imakefile

This only seems to be relevant to X.org – this defines if your should inherit paper size from C or en_US. Add your locale and its permutations to the correct list.

New Locales: known problems

1.12.6 gdm

Your locale works in any terminal you open from the desktop, but won't be used in the desktop itself. This happens because of a copy of locale.alias that gdm holds in /etc/X11/gdm (at least in gentoo).

Add you locale to this file or it will be ignored by gdm.

- *All about locales*
- *Resource for locale data content*
- Creating locale files
  - *Glibc locale files*
  - OpenOffice.org
  - CLDR
  - *KDE*
  - Solaris
- *Patching X11 to accept your locale files*

1.13 Tools

1.13.1 Translation Tools

Using PO editing tools

You need a PO editor to edit PO files which is the main format used by Free Software, it is also the format the the Translate Toolkit uses when converting Mozilla and OpenOffice.org files for editing

There are a few PO editing tools available. Your choice really depends on which platform you are running:

- I use Windows
  - Install Virtaal, *Poedit*, or IniTranslator
- I use Linux
  - I use KDE
    - Install or use Lokalize or Virtaal
  - I use Gnome
    - Install or use Virtaal or GTranslator
- I am not allowed to install software on my computer
  - If possible use a *web-based* translation tool. Not all translation projects have a web interface for translation. Possible options are Pootle, Rosetta and Kartouche. POEditor is available with a graphic interface.
- I am not allowed to use the web
  - If you cannot install software or use the web then you might want to look at using CSV files edited in a spreadsheet. You will have to have access to email though.

Other alternatives

XLIFF report – 1 October 2007

(investigation by Samuel Murray)

General overview

I've investigated some of the programs currently capable of exporting/importing and translating XLIFF files, and here are my findings.

- Some localisation programs like Sisulizer, Catalyst and appTranslator can export to XLIFF, and import it again, but I get the impression that these are vendor-specific dialects of XLIFF which would be dangerous to roundtrip via the Translate-toolkit. For example, roundtripping an appTranslator XLIFF file loses a lot of information.
- Some CMSes claim to export/import their content in XLIFF. Drupal is one, but I haven't been able to get my hands on exported XLIFF from it.
- Some CAT (machine aided human translation) tools use XLIFF as their internal format. Examples are a concept version of OmegaT called OmegaT2, Cafetran and Transolution. I haven't been able to these programs working to see what their XLIFF formats look like.

SDL Trados 2007

- You can use Trados to translate Translate-toolkit's XLIFF files
- Trados doesn't have a real XLIFF filter; it treats XLIFF as XML, but it is smart enough to protect the source text
- The file must be sourcecopied (you can do that using msgen)
- To translate, simply open the XLIFF file in TagEditor.

OmegaT 2.1.1

- OmegaT doesn't have a real XLIFF filter; it simply treats XLIFF as XML
- You can probably use OmegaT to translate Translate-toolkit's XLIFF files
- The file must be sourcecopied (you can do that using msgen)
- The file extension must be XLF
- Previously, OmegaT saved XML files with LF even if the source file is CRLF (not sure if it still does that).

SDLX (build 7014)

- SDLX doesn't seem to have a real XLIFF filter; it converts XLIFF to its own ITD format
- You can use SDLX to translate Translate-toolkit's XLIFF files (but see warning)
- The file extension must be XLF

Warning: Empty target tags in SDLX will be displayed as <0/>, and these must be deleted before saving as XLIFF again, otherwise they are included in the XLIFF file. For this reason I'm not sure how SDLX will handle more complex XLIFF files. Also note that SDLX overwrites the original XLIFF file without asking.

DVX (DejaVu version 4 aka version 'X')

- DVX doesn't have a real XLIFF filter; it treats XLIFF as XML, but you need to create the filter yourself using the particular XLIFF DTD and a filter creation utility inside DVX.
- I think you can use DVX to translate Translate-toolkit's XLIFF files (untested)

- The file must be sourcecopied (you can do that using msgen)

Heartsome XLIFF Editor 6.2-10 Build 7D7-3-1B

- You can use Heartsome XLIFF Editor to translate Translate-toolkit's XLIFF files
- It saves the file with LF line endings, even if the original file had CRLF line endings.
- It adds deprecated prop-group tags, but those don't affect the Translate-toolkit

Open Language Tools 1.2.7

- OLT doesn't recognise XLIFF files by Translate-toolkit
- It only supports XLIFF 1.0, and reports XLIFF 1.1 and 1.2 as invalid.

You may be able to use a text editor to edit PO files, since they are text:

- **vi** – You can use the vi editor. It includes syntax highlighting, but you might want to check if there is a newer version
- **emacs** – The Emacs editor has a PO mode for editing PO files.

Or you can try to convert and/or use another format:

- CSV – You can use a Spreadsheet to edit PO files that have been converted into CSV. It works quite well but is not highly recommended if you have good access to a PO editor listed above.
- XLIFF – This is an emerging translation interchange standard. We will see much more available in this format, convertors are being created to move PO files to Xliff and a number of editors both GPL and commercial are being made available. A short report on *XLIFF support for MS Windows*.

Feature comparison of translation tools

This table provides feature comparison among various translation tools, both Free (FOSS) and commercial.

	Virtaal	Wordforge	*Poedit*	G-Translator	Lokalize	*KBabel*	OLT XLIFF Editor	*Loc Factory Editor*	Emacs PO mode	Transolution	OmegaT	Wordfast	Swordfish	Heartsome
FOSS	✓	✓	✓	✓	✓	✓	✓	–	✓	✓	✓	–	–	–
Active	✓	✓	✓	✓	✓	–	✓	✓	✓	–	✓	✓	✓	✓
Implementation	Python/GTK+	Python/Qt	C++/wxWidgets	C/GNOME	C++/Qt/KDE	C++/Qt/KDE	Java/Swing	Java/Cocoa	Elisp	Python/GTK	Java/Swing	*?*	Java	Java/Swing
Platform														
Windows	✓	✓	✓	–	✓	*unofficial*	✓	–	✓	✓	✓	✓	✓	✓
Linux	✓	✓	✓	✓	✓	✓	✓	–	✓	✓	✓	–	✓	✓
MacOS X	*?*	*in X11*	*in X11*	–	*in X11*	*in X11*	✓	*pre-10.6*	✓	*?*	✓	✓	✓	✓
Supported formats														
PO	✓	✓	✓	✓	✓	✓	✓	✓	✓	–	✓	–	✓	–
XLIFF	✓	✓	–	–	✓	–	✓	✓	–	✓	–	–	✓	✓
TBX	✓	✓	–	–	✓	–	–	*?*	–	–	*?*	*Pro (import)*	✓	✓
Qt Linguist(.ts)	✓	–	–	–	–	–	–	–	–	–	–	–	✓	–
Qt Phrasebook(.qph)	✓	–	–	–	–	–	–	–	–	–	–	–	–	–
Translation memory														
TMX	✓	✓	–	*?*	✓	✓	–	✓	–	*?*	✓	✓	✓	✓
WordFast TM (.txt)	✓	–	–	–	–	–	–	–	–	–	–	✓	–	–
Internal format	*?*	*?*	✓	*?*	–	–	✓	*?*	*?*	*?*	*?*	✓	✓	*?*

Available translation tools

This is a list of translation editors that are useful for translating Free and Open Source software. See the *comparison chart* if you need to evaluate features. We have included both Free and non-Free software but you will find the Free more extensively covered as this is what people are actually using to translate Free Software.

OmegaT

This is a Java based tool for translating HTML, OpenOffice.org and plain text documents. We would recommend that you use it for translation of these documents when found in Free Software projects.

OmegaT can also translate key=value files, Java .properties files and a number of other files, including Microsoft Office 2007 (with tag soup).

Resources

- Home Page

Issues

- OmegaT's handling of PO files is non-standard, but it can handle partially translated PO files. The dev team is quick to fix any reported bugs.
- OmegaT's support for msgctxt is not 100% right.
- OmegaT's support for plurals may not be 100% right.
- OmegaT does not display comments to the translator (including translator comments and context comments).
- OmegaT translates identical source segments with identical targets.

LocFactoryEditor

the XLIFF and PO editor for Mac OSX

LocFactoryEditor, or LFE for short ;) is a reasonable recent entry to the translation arena. Originally intended for the professional translation market, it has been extended to handle PO files, and is in fact available free as a dedicated PO editor.

Mike of Triplespin has worked closely with at least one open-source translator (me ;)) to provide a really solid and indispensable translation tool, not only for volunteer translators, but for any translator using Mac OSX.

His aim is to create a full toolset for Mac OSX translators, firmly based on the current standards.

I've spent a lot of time testing and using his software, and I recommend it strongly. I'm impressed with a lot of things: the standards support, the built-in XLIFF/PO conversion, the TMX/Compendium translation memory, etc., but the one central thing that grabs me when I work in LFE is the interface.

It's intuitive. Nothing crowds the eye, and everything you need is right there, plus keyboard shortcuts if you prefer (and, of course, in OSX you can assign your own as well ;)).

Smart folders mean simply one click to show **only** the untranslated or fuzzy strings, and counters at the top of the page alert you if spaces or symbols aren't consistent between original and translated string.

The Notes facility makes sense of the stodge of PO headers, and allows you to set your own Notes, input them with shortcuts, and edit them directly. Everything converted neatly back into PO format in the output file.

We have plurals support, of course, can show or hide obsolete strings, and LFE maintains a backup file automatically (to protect you from yourself ;)).

If you also translate Mac applications via AppleGlot, LFE handles the entire process, from first pass to final test, work glossaries and .loc files.

Try it. The PO-only version won't cost you anything, and it'll save you a lot of time and hassle. I bought the full version, which is very affordable, and have had my money back in sheer features and convenience many times over.

As a Mac OSX translator, I had several options:

- emacs — very cluttered interface and (to me) non-intuitive operation: drove me around the bend
- kbabel — finally installed it via fink after months of struggling with error messages and forums/mailing lists, and X11 for Mac OSX won't input Vietnamese … !!
- gtranslator — wasn't able to install it, despite many tries

I also use Pootle online, but in general and most heart-feltedly, I give thanks daily for LFE!

Clytie Siddall

Vietnamese translator for open-source projects

Poedit

Poedit is a PO editing tool that will run on Linux, Mac or Windows. It has a reasonably simple interface as well as translation memory. Its catalogue manager does not seem logical at first but is powerful in that it allows you to manage multiple projects.

Virtaal can import settings and translation memory from Poedit.

Issues

- Poedit knows a file's encoding from the PO header. Therefore, if a PO file has no header, Poedit opens a file using the system's default encoding. This means that if you're using Windows, and you're opening UTF-8 PO files without headers, Poedit will open them in Windows plaintext encoding, and so break the files.
- It can only edit PO files.

Resource

- Home page
- Download

Using Poedit

Setup

When you first run Poedit you will be asked for some information. These items are stored in the header of the PO files and include your name, email address, etc. The tabbed dialogue that appears needs the following information:

- Personalize: Supply your name and email address
- Editor: you can safely leave these unchanged
- Translation memory: leave unchanged for now
- Parser: unchanged

Click OK and you are presented with the PO editing interface.

The first time with Poedit

Once Poedit is started you will see the translation interface.

Click file open to open and existing PO file. If you have no PO file or only a POT file. Then simply copy the POT file so that it ends in .po

With a PO file open you will see all strings in the top part of the interface. Strings are shown in different colours:

Cyan	untranslated
Yellow	fuzzy – needs to be examined and might need to be retranslated
White	translated

You translate in the lower half of the interface. The original string or English string appears in the left. You type your translations in the bottom right hand section. It seems that keyboard navigation does not work properly so you will have to select each new string with the mouse.

Don't forget to save regularly using `Ctrl+S`.

Press `F1` at anytime for the Poedit user manual.

Using the Catalogue Manager

The catalogue manager (*File* → *Catalog manager*) allows you to manage projects with multiple PO files.

Click on create new translation project. Then for each directory that has PO files add it to the project.

Using Translation Memory

FIXME

KBabel

KBabel is a KDE 3 application for editing PO files. A Windows port was started, but apparently never released. In KDE 4, KBabel was replaced by Lokalize.

Virtaal can import settings and translation memory from KBabel.

Installing

- Linux – Forms part of the kdesdk package. Install that and any dependencies.
- Windows – Porting still in progress (screenshot)

Using KBabel

First Time

FIXME

The Editor

FIXME

Catalogue Manager

FIXME

Ini Translator

Ini Translator is a utility program to translate ini-style language files and has a look and feel reminiscent of *Poedit*. There are several plugins supplied with the install including parsers for Gettext PO format and others.

Resources

- Home page

Multi-format editors

- Virtaal
- WordForge Editor (formerly Pootling)
- Lokalize
- *OmegaT*

non-FOSS

- Heartsome Translation Studio
- Swordfish
- Wordfast
- *LocFactoryEditor*

PO editors

- *Poedit*
- GTranslator
- *KBabel* (replaced by Lokalize in KDE 4)

XLIFF editors

- OLT XLIFF Editor
- Transolution

Qt TS editors

- Qt Linguist

XLIFF converters

- file2xliff4j – converts various file formats to XLIFF (written in Java)

Other Translation editors

- *Ini Translator*

Web-based tools

- Pootle
- Rosetta (proprietary, used for Ubuntu linux)
- IRMA, previously used at Linspire

Online Translation Tools

There are a number of online translation tools that you might want to use to assist your translation projects. Being online they allow you to work from anywhere, it also allows many people to participate without having to install software or configure a PO editing tool.

As the developers of Pootle we are happy to see that the idea of online translation has become more popular since we originally gave Pootle to the world.

Web-based Tools

These are some tools of which only some are still available. The others are kept for historical reference.

An important issue for these is the extent to which they allow offline translation, access control and quality assurance.

- Pootle – A multi-format system for translation and translation project management with many features. Actively being developed and used by Translate.org.za along with the Translate Toolkit and Virtaal. There is also an active translation portal for those who cannot host there own copy of Pootle. **Pootle is released under the GPL.**
- Kartouche – Developed for the Welsh localisation team. See also Omnivore their online compendium tool. See the Welsh team's live site.
- IRMA – Developed by Linspire aiming to make things as simple as possible for translators. Michael Robertson of Linspire talks about his first date with IRMA. **This is not free software.**
- Rosetta – Developed by Canonical for the translation of Ubuntu. Aims to make it easy to translate software, but is hard to install. **Used to be proprietary, but now under AGPL and other licenses.**
- WeBabel – A PHP & MySQL translation portal. **Released under the GPL**
- An OpenOffice.org specific tool – Used by the Danish team and developed by Søren Thing Pedersen
- Entrans – A web based translation system originally for Indian translation. GPL.

- Transifex – A web based front end for version control systems with an editing interface. GPL, but demands the signing of a Contributor License Agreement to contribute
- Weblate - A free software web-based translation tool with tight version control integration. It features a simple and clean user interface, propagation of translations across components, quality checks and automatic linking to source files.

Using CSV and your Spreadsheet to edit translations

A CSV (Comma Separated Value) file is one in which each field or cell is separated by a comma (or another character) and each record or line is separated by a newline.

When using CSV files generated by po2csv you will find the data as follows:

Column	Description
A	contains the location information from the PO file. This is sometimes useful for working out what the message is used for as its location and the name of the message can provide clues.
B	contains the original or English text
C	you add your translations to this column

Notes

Preserve filenames, do not change the case of the filename. Return files in a ZIP archive in the same order and directory structure. This is important because the reverse process, converting CSV to PO is made easier if you do not change these.

How to open a CSV file

When opening a CSV file you will be presented with a number of questions. Here are the answers you should provide:

OpenOffice.org 2.0

- In the text import dialog. Ensure that the character set in 'Unicode UTF-8'. Under 'Separator options' ensure that the 'Separated by' radio button is selected and the 'Comma' is checked. The text delimiter should be " (double quotes).
- Check to see that the import wizard has identified the 3 columns correctly
- Click OK

Microsoft Excel 97/2000

There is a problem with Microsoft Excel 97 and 2000 in that it opens a UTF-8 file as if it is ASCII. Excel does save CSV files correctly, if the encoding of the opened file is retained, but Excel by default does not retain the encoding. Some success has been had using XSLGEN (edit the xslgen.bat file to reflect the location of your installation of Excel) to open UTF-8 CSV files correctly in Excel.

Getting the most from your spreadsheet

Initially the first column will stretch across the page as it contains a large amount of data, but you cannot see or work with this data. Therefore do the following:

- **Adjust the column widths** – 5” works well as you can then see both the original and translation clearly.
- **Adjust the cell formatting** for the first column to allow the insertion of automatic line breaks (this will cause the data to wrap within the column and make it all readable) this also allows for automatic hyphenation which will then hyphenate long words. The result of this step is that you can now read all data in column one as the rows will become larger to accommodate this information.
- You can repeat this process for other columns but then messages will wrap differently to how they appear in the translation. Do this only if you really don’t have space or otherwise look at text shrinking options.

Each Spreadsheet application will achieve the above in a slightly different way. Here are specific instructions.

OpenOffice.org

- Right click on the header of Column A. Select “Column Width”. Type in a value in the “Width” field.
- Right click on the header of Column A. Select “Format Cells…”, then select the “Alignment” tab. Now under Properties enable “Wrap text automatically” and “Hyphenation active”.
- If you can’t see column B or C clearly. Select the header of the column. Select “Format Cells…”, then select the “Alignment” tab. Now under Properties disable “Wrap text automatically” and enable “Shrink to fit cell size”

Microsoft Excel

FIXME

- Choosing a *translation editor*
- *Comparison Chart*
- *A list of available tools*
- *Online or web-based tools*
- *Using your spreadsheet to translate*

1.13.2 Glossary Tools

Glossary Tools

For a quality translation it is important to develop a good glossary of terms. In this section we list the various tools that can help you create and manage such a glossary.

Glossary management

To create a glossary of terms (words and phrases), you can either create it as you go along during a pilot project, or you can extract terms from the source text and compile them into a glossary.

The advantage of creating it during a pilot project is that the term list is created in context, but the disadvantage is that it is a slower method and it yields less terms (although most terms are useful). The advantages of automatic term extraction are that it is fast and yields potentially more relevant terms. The disadvantage is that terms are not viewed in context, and you have to initially search through many unimportant or useless terms.

The disadvantages of either method can be overcome, though.

Bulk term extractors

Bilingual term extractors

Since some texts are available in bitext format (which can be a TM or a PO file or similar), it would be possible through smart guessing to figure out which source words fit which target words. These tools are generally called "word aligners" in the computational linguistics industry, and sadly most "free" tools are free for academic purposes only. Word aligners have great potential because bitexts are usually translated by humans, and so the terms are likely to be accurate.

You might also be able to do monolingual term extraction on a TM.

Poterminology

The Translate Toolkit contains poterminology which can do bilingual term extraction. It does frequency analysis on the input files, and uses stop words to improve the results. It has several parameters that allow you to change its behaviour. It does some alignment and fills in all translations for a term that it found in the translation files.

Monolingual term extractors

Monolingual term extractors usually look for words or phrases that occur more than X number of times in the text, or for words that occur in the text but do not occur in a dictionary or established term list. Such an extraction usually contains a large percentage of useless terms unless steps are taken to remove irrelevant repetitions from the result.

Poterminology for monolingual extraction

poterminology (see above) can also work on untranslated files, in which case it would simply be doing monolingual term extraction.

ExtPhr32 by Tim Craven

Tim Craven's ExtPhr32 for MS Windows is not GPL but it is freeware for all purposes. It is very fast. Unfortunately it converts all terms to uppercase. You can also use a stoplist. You can choose how many occurrences of a term must be the minimum, and you can choose the minimum number of words in a term. The output can be exported to two column plaintext (the second column contains a count).

PlusTools from Wordfast

PlusTools is a macro that runs within MS Word on MS Windows. It is not GPL but it is freeware for all purposes. It is slow but potentially useful for smaller texts because it can exclude words that occur in MS Word's spellchecker and/or thesaurus, or words that have less than X synonyms in the thesaurus. You can also exclude certain words (similar to a stoplist, but you can add any words to it), words beginning with a certain set of characters, and words that are smaller than X number of characters.

Other tools

- http://uplug.sourceforge.net/

GPL collection of corpus tools.

- SDL MultiTerm 7 Extract Freelance

Extracts terms from Trados compliant bitexts. Pricetag: EUR 525.00 per single user licence.

Concordancers

A concordancer is a tool that displays a word in context. An excellent, easy to use concordancer, is Corpsis (previously Tenka Text). Or, if you're willing to pay some money, try Mike Scott's Wordsmith tools. Corpsis can display multiwords phrases using wildcards.

http://corpsis.sourceforge.net/

Glossary editors

Whichever way you look at it, a glossary is a database, and most comprehensive glossaries can be edited in a database editor tool. For simple, three-column glossaries, a spreadsheet program may be all that's necessary, though.

Virtaal can be used to edit many formats, including many formats commonly used for storing terminology.

(todo)

Glossary viewers

There are quite a few glossary viewers, but they often require that you convert your glossary to their weird format. Examples are StarDict, Jalingo, jDictionary and Pododict. If you're willing to pay for a non-free product, AnyLexic is an excellent glossary tool that supports simple formats too.

Virtaal can open many formats, including many formats commonly used for storing terminology.

(todo)

Web based creation and management tools

- Pootle can do terminology extraction based on poterminology since version 2.1. Term lists can be translated and downloaded for offline editing/use.
- KiPot – produced by it46 and used by the Kiswahili Localisation team
- Gakartuleba.org – custom tool used by the Georgian localisation team (requires free account)
- Glossword – helps to create your own dictionary or dictionaries
- *Glossary creation and management*
- TranslateIt on OSX provides system-wide translation into single or multiple languages, as well as glossary creation

1.13.3 Source Control

A basic guide to CVS

CVS is a version control system. It keeps track of files, changes and who made the changes. Some projects are starting to use Subversion for managing their files.

Resources

- Crash course
- Full course

Tortoise CVS

This is a Windows based application for using the CVS version control system. CVS is used by many localisation projects and eventually you will probably need to interact with it.

It is also useful if you can localise but only on a Windows based machine at work or University. This way you can still contribute to FOSS localisation on from your Windows machine.

See also *Win CVS* as an alternative to Tortoise CVS.

Resources

- Tortoise CVS home page
- Tortoise CVS download page

These pages from the Fedora project give nice instructions on how to setup and use TortoiseCVS.

- Downloading Tortoise CVS
- Using Tortoise CVS

Win CVS

Some people recommend WinCVS over *Tortoise* CVS. If you are a user of either then please give us your feedback.

Resources

- WinCVS home page
- Downloads
- Screenshots
- Documentation

SVN Tips for Translators

SVN is one of the most useful tools available for keeping up-to-date with your PO files. It's not difficult to use, and once you get used to it, you'll wonder how you ever got on without it. ;)

XXX

Why use Source Control?

Most projects use some form of source control to manage their files. Source control makes it possible for many people to work on the same file, without confusion or lost data. Anyone who wants to change that file simply gets the latest copy from the **repository** (location of the shared files), adds his or her improvements, then **commits** (writes) the file back to the **repository**. The source control software makes sure that each person's additions are integrated in the main file.

For example, if you add a section on Kumquats to the document on citrus fruits, then commit it, your colleague will see it when she **checks out** (downloads from the repository) the latest copy, to add her views on Grapefruit (good for you but sour comfort in the mornings). Once her grapefruit information is committed, another colleague may add forceful comments on the delightful taste of grapefruit, and commit them. The original non-lover of grapefruit can **check out** the latest file and add aspersions on her colleague's tastebuds, which were shot off in the war. All nice, clean fun, while the SVN server collects all the information and maintains the current file.

How does this affect us translators? **Source control means we have access to the latest .po file at any time.**

Note: a GUI front-end for SVN like svnX makes using SVN even easier! Your file-browser might also have embedded SVN functions.//

How do I get the files?

All you need to get your latest PO file, is the source control **address** of its directory on the server. In the case of SVN, the address will either look like svn://svn.servername.org/path/to/directory/po or https://svn.servername.org/path/to/directory/po.

In other words, SVN files, just like webpages, have their own address. Like many websites, they ask you to login. If you want to be able to **commit** files (add them to the **repository**) you need permission (**write access**), just as you would need permission to add or change files on a website. This permission system protects our hard work from being altered or damaged by unauthorized people.

For example, Sourceforge uses the https://svn.servername.org method, which requires you to input your Sourceforge account username and password. You would also need permission from the developers of your Sourceforge project, to **commit** files. Many svn://svn.servername.org servers use SSH to identify you when you connect. A server may be willing to let you **checkout** (download) files anonymously, but it certainly doesn't want to accept files from a stranger. Don't be a stranger: setup your SVN access. :)

To commit your files, if you don't have write access, you will be asked to upload them using a tracker, or to email them to the developers, who will later commit your files to the repository.

What are the key points?

- **Checkout** your **working copy** (your own local copy of the files on the server). This might be the whole **source tree**, containing all the files making up the project, or it might be part of that tree, only one **branch**, or only the `/po` directory containing your PO file. Checking out directories (you can't checkout single files) is as simple as **svn co svn_address**. You can use the whole word **checkout** or simply **co** for short. Type the command when you are in the directory where you want the server files to arrive.
- Notice an **.svn** directory that has also arrived from the server. This little folder is the key to source control. It keeps notes on where your files belong on the server, and what **revision** (version) of the files you have. Every time you use SVN in that directory, that little **.svn** folder updates its information. It knows if your files come from a **branch** (separate version of the program), it knows exactly when you last updated or committed them. So anything you do outside your **working copy** doesn't count: you must use SVN within it.

- Always **update** your **working copy** before making any changes. Updating asks the server for the latest **revision** of the file. If you run **svn update** in your `/po` directory, the server will update every file in it to the latest **revision**. Your translation editor or text editor may also be able to update individual translation files via SVN. Either way, you must update your `xx.po` from inside its `/po` directory (or higher((Inside the **working copy**, SVN commands have a hierarchical effect: running **svn update** in any directory which contains `/po` will also update `/po`, and any other contained directories. So, if `/po` is on the path `/trunk/messages/po`, updating the `messages` directory will also update `/po` and any other directories it contains, and updating `trunk` will update the whole working copy.))). Even if you have downloaded a new copy of the file from the Web, or have been sent a new copy by email, you must update your working copy of `xx.po` before making any changes. Otherwise the server won't know that you have the latest file.

Problem-solving

Problem 1

Minh receives the latest `vi.po` file for Program X by email, or downloads it from the Web. He saves it to his `/po` directory. He updates his translation. Then he tries to commit the updated PO file. The SVN server refuses it, says it is out-of-date! Why? It's the latest file.

Solution 1

The file may be the latest, but the SVN server doesn't know that. As far as it knows, Minh last updated his `vi.po` file via SVN some time ago. He certainly hasn't updated his `/po` directory from the server. So the server refuses any files from what it sees as an old working copy. Imagine this conversation:

vi.po: Let me in!

Program X svn server: Who are you?

vi.po: I'm, um, vi.po.

Program X svn server: So you say. Are you the latest vi.po? I don't want you overwriting a newer file. I have to check up on you. Wait over there, and don't interrupt.

***Program X svn server** queries the .svn folder in the same directory with vi.po*

Program X svn server: Do you know this vi.po?

.svn folder: There's a vi.po here with me.

Program X svn server: How recent is it?

.svn folder: This directory hasn't been updated from the server for a couple of days.

Program X svn server: Really? thanks.

***Program X svn server** now has the information it needs.*

Program X svn server: Hey, vi.po!

vi.po: Yes? Can I come in now?

Program X svn server: Get lost. You're an old version.

vi.po: But I'm not! I'm the latest version, all updated!

Program X svn server: Not according to your .svn directory. End of conversation. Stop wasting my time.

Doing it right 1

Minh receives an updated `vi.po` file via email, or downloads it from the Web. He saves it somewhere else on his disk (*not* in his working copy `/po` directory), then updates his translation. When he's ready to commit the file, he runs **svn update** in the `/po` directory where the old file is. *Then*, if he is sure his edited file is the latest one, he saves it on top of the updated one from the server. He runs **svn commit** inside the `/po` directory.

Problem 2

Sonja wants to know if her `ro.po` file for Program Y has been updated or not. She looks at the `ro.po` file in her working copy `/po` directory. She updated it yesterday, so it's pretty current, she thinks. It has some untranslated and fuzzy strings, so she updates the translation, then commits the file. The SVN server rejects the file, saying it is out-of-date. Why? She updated it yesterday!

Solution 2

Another member of Sonja's translation team has fixed some typos in the existing translations. He committed his changes this morning. So yesterday's file is no longer the current copy of the `ro.po` file.

Doing it right 2

Sonja wants to know if her `ro.po` file for Program Y has been updated or not. She updated it yesterday, but who knows what has happened on the server since then? She runs **svn update** in her `/po` directory. She now has the current copy of `ro.po`, so she can update her translation, then commit it.

Problem 3

Jean-Christophe wants to know if his `fr.po` file for Program Z has been updated or not. He updated it yesterday, and in our example, he is unfortunately the only member of his translation team, so there aren't any other translators! Nobody else would change the file. So he goes ahead and updates the translation, then tries to commit the file. The SVN server rejects the file, saying it is out-of-date. Why? No other translator can have changed the file!

Solution 3

The developer added a couple of new strings, and committed them this morning. She then updated all the .po files. Jean-Christophe's version of the file is no longer the latest one. The moral of this story being: you can never assume you're the only person who would change a file, unless you're the only person with write access, and even then, you might have setup some regular scripts which update the files! So don't assume: **svn update**.

Doing it right 3

Jean-Christophe wants to know if his `fr.po` file for Program Z has been updated or not. He updated it yesterday, but anything could have happened since then. He runs **svn update** in the `/po` directory. He now has the current copy of `fr.po`, so he can update his translation, then commit it.

Conflicts

There is still a small probability that someone else might change the file on the server *while you're editing the file*. It's unusual, but it does happen, either by coincidence, or in very busy projects/teams. If you've svn-updated the file before working on the translation, and your commit is rejected as being out-of-date, you can do one of two things.

Solution

- Run **svn update** in the `/po` directory. This will create **conflicts** between the newer file on the server and your file. Extra files called `xx.po`.mine and `xx.po`.zzzz (where zzzz is the revision number of the current file on the server) and possible another dot file at the top, will be created in the `/po` directory. The **conflicts** will be shown between markers (>>>>> and <<<<<) in your `xx.po` file, with both copies of each changed line or string shown: marked **mine** and **zzzz**. You can choose which one to keep in each case. Make sure you have removed all signs of **conflict**, including the extra files, before trying to commit again. This is a messy process, and should be avoided if possible.
- If you have a backup copy of your completed translation (and you should *always* keep a backup copy of your work, asking your translation editor to create one automatically, or creating one in another directory before trying to commit), delete your `xx.po` file in your `/po` directory. Run **svn update** in the `/po` directory. The server will **update** the directory to the current revision by making sure you have *all* the latest files, including a brand-new, shiny copy of `xx.po`. (It will replace any missing files.) Now you can save your backup copy over the new `xx.po`, or update your translation in that file, whichever works best for you at the time. Then commit! This method is much safer, since it avoids the messy **conflicts**. When you get into a mess with your working copy, you can always delete your own files and **update** them from the server. **svn update** will always **checkout** the current copy of that directory for you.

Summary

So, basically you can **checkout** your **working copy** (at least the directory containing your file), which gives you your `xx.po` file and the .svn folder which keeps track of its status. Anytime you want to know if the file has changed, run **svn update** in that directory. Then open `xx.po` in your editor, add new translations or fix older ones, save it, and **commit** it! Don't forget to edit the **ChangeLog**[1] if there is one.

Footnote

- Using *A basic guide to CVS*
  - *Tortoise CVS* and *Win CVS* – CVS applications for Windows
  - CrossVC – cross-platform CVS GUI front-end
  - The BBEdit text editor for OSX integrates CVS
- Using SVN
  - An SVN Summary for people switching from CVS
  - *SVN Tips for Translators*
  - svnX – intuitive SVN front-end for Mac OSX
  - PathFinder for OSX has an SVN pane, and a multi-tab terminal pane
  - The BBEdit text editor for OSX integrates SVN

[1] A text file, usually in the same directory, which contains entries of the format `Date Filename What changed Name and Email address`.

- Using Git
  - A Git-SVN Crash Course for people switching from SVN
  - How to use Git at Debian
  - How to use Git at GNOME
- Using Bazaar
  - The Bzr Wiki
  - A Bzr tutorial
  - Bzr for SVN users
- Using SSH
  - SSHKeychain on OSX manages your SSH keys
  - SSHAgent on OSX allows seamless SSH logins
  - SSH Tunnel Manager on OSX manages your SSH tunnels (e.g. to OpenOffice.org)

1.13.4 Other tools

Gettext Hacks

A few useful Gettext hacks.

Branch merge

Needs to merge translations from a branch back onto HEAD?

```
msgmerge -C HEAD.po BRANCH.po HEAD.pot > NEW-HEAD.po
mv NEW-HEAD.po HEAD.po
```

Use *HEAD.po* as a compendium to make sure you get any translations that might have been done in HEAD. Then migrates *BRANCH.po* to be up-to-date with *HEAD.pot*. The output *NEW-HEAD.po* should then be moved to *HEAD.po* if the merge happened as planned.

Generating Statistics for your project

It is always nice to track statistics for your projects.

- Pootle presents statistics for all hosted files
- pocount can output word and string statistics for files (can also output to CSV files for use in a spreadsheet)
- Czech OpenOffice.org teams stats tool
- KDE statistics pages
- Kumulate

Czech OpenOffice.org teams stats tool

This is the tool used by the Czech language team for OpenOffice.org:

- http://tmp.janik.cz/OpenOffice.org/l10n/PO2STAT
- The current Czech stats page

In the words of the author, Pavel Janik, this requires no "*database, PHP and similar ugly stuff*"

KDE statistics pages

FIXME track down the PHP scripts for this

This seems to be the original stats generation page as most others look frighteningly similar to the KDE pages:

- http://i18n.kde.org/stats/gui/trunk/index.php

UPDATE 24.10.2009

The scripts for the KDE statistics page as seen here:

- http://i18n.kde.org/stats/gui/trunk-kde4/team/

Are generated using the l10n-stats scripts, which can be obtained here:

- http://i18n.kde.org/about-stats.php

Kumulate

A browser-based visualiser for translation trees. The code is released under the GPL.

- http://www.dotmon.com/kumulate/

Although PO files will show us which strings have changed, sometimes the string can be very long and complex, such as a help screen. Also, HTML and text files do not show changes in the organized, summary way gettext PO files do in our translation editors.

How to find specific changes

Text editors, and some translation editors, will run a "diff" routine which compares two directories, and shows you which files have changes. That's certainly a good start.

But how do we find specific changes? Ordinarily, we can run "diff", in the terminal, or in our editor:

```
diff -u old_file new_file
```

and the output will show in our terminal. To output the information to a file:

```
diff -u old_file new_file >output_file.diff
```

But this will only show what the editor regards as line-by-line changes, which in a PO file are often whole sections.

You can hard-wrap the older file (but not the newer file) before running diff, and that will actually give you a line-by-line comparison (you can remove the line-breaks from the older file afterwards, if necessary). That's a definite improvement.

But we can do even better.

If you run **wdiff** instead:

```
wdiff -u old_file new_file >output_file.diff
```

you get output that looks like this:

```
Test of a very long line where something [-has changed-] {+did change+} and
you don't know what it {+really+} was, why?
```

In other words, you get a **word-by-word comparison**.

You can clearly see what has changed: the {braces} show additions, and the [brackets] show deletions.

If you use the 'less' program in the terminal:

```
wdiff -l  testA testB |less
```

you will also see bold text and underscored text highlighting the differences.

You can find more information on wdiff.

GUI software

Winmerge (MS Windows)

http://winmerge.org/

To compare two files, drag and drop one of them onto the Winmerge icon, and Winmerge will prompt you for the other file.

You can use Winmerge in combination with Tortoise SVN. In the SVN tree, right-click the folder with changes in it, and select *Tortoise* → *Check for Modifications*. In the Modifications dialog, where modified files are displayed, simply double-click a file to have it opened in Winmerge.

You can make changes to a file in Winmerge.

Winmerge requires a BOM to recognise a Unicode file, so if your Unicode files don't have BOMs (which is the case with Toolkit produced files), you have to tell Winmerge to "fall back" to Unicode when opening a "non-Unicode" file (no, I'm not joking). Go *Edit* → *Options* → *Codepage* → *Custom Codepage* and type in 65001 (which is the codepage for UTF8).

Winmerge can also be used to compare trees and directories.

- *Gettext Hacks*
- *Generating Statistics for your project* generation
- *Comparing Files* – how to find changes in translations (diffing)

1.13.5 CAT tools used by translators

Common CAT tools

Translators often translated documents using so-called CAT (computer aided translation) tools. Some of the more common CAT tools include Wordfast, OmegaT, Trados, Déja Vù, Metatexis, Isometry, and others. These tools are typically used to translate ordinary documents, but some of them can also be used to translate l10n formats like PO, CSV, XLIFF and other bilingual formats.

There are two types of CAT tools, namely those that work directly on the source text, and those that work indirectly on the source text by extracting the text from it.

Some CAT tools

OmegaT

OmegaT is written in Java and is GPL. It can translate OpenDocument files, well-formed XHTML, Java .properties files, key=value files and plaintext files, as well as a number of other formats. OmegaT extracts the text from the source documents, and the translator translates the text within the OmegaT environment. Text formatting is dealt with through special OmegaT tags.

Déja Vù

Déja Vù is propriatory software (EUR 990 per licence). It can translate many other propriatory formats and open formats, including Gettext PO. It works, similar to OmegaT, by extracting text from source files so that the translator works within the program's own environment.

The disadvantage of programs like Déja Vù and OmegaT is that they can only handle existing, well-known text formats. The advantage is that they usually handle these formats very well or comprehensively.

Wordfast

Wordfast is a Visual Basic macro that runs inside Microsoft Word, and it is propriatory (EUR 250 per licence). It translates any file that can be opened in Microsoft Word, Excel and PowerPoint. It does not extract text, but instead it selectively allows the user to translate the portions of the file that needs to be translated.

Trados (bilingual RTF)

The way Trados handles files, is the same as the way Wordfast does it. In fact, Trados and Wordfast can read each other's files – that is, the bilingual RTF files. This is the file type that is most useful for new or rare file formats. Trados is proprietary

Trados (TagEditor, TTX)

A newer method used by Trados is called TagEditor (it produces TTX files). TTX files are bilingual XML files that are opened translated in TagEditor, which is basically a user-friendly XML-viewer (but it can only view TTX XML files). In this sense TTX is similar to XLIFF or PO, because it is a bilingual file which is generated from the original file.

Features of CAT tools

CAT tools generally have additional features prized by translators, such as the ability to capture and use a translation memory, automatic on-the-fly fuzzy matching, automatic glossry recognition, various keyboard shortcuts to improve speed, and power reference search facilities.

Introduction to Wordfast

This page is an overview of Wordfast, specifically with PO and l10n translation in mind. For more comprehensive detail about Wordfast, read the user manual, downloadable from the Wordfast web site.

The way Wordfast translates

Since Wordfast is simply a Visual Basic macro that runs inside Microsoft Office, translators using Wordfast are in essence using Microsoft Office and all of its available features. Wordfast can therefore be used to translate any document that can be opened in Excel (the spreadsheet) or Word (the word processor). Wordfast can't translate binary files, but it can translate text files in various encodings.

Translators translating any plaintext file would do the following:

1. Open the text file in Word, selecting the appropriate text encoding, eg UTF-8
2. Save the file as an RTF or MS Word document
3. Prepare the file for translation (optional)
4. Translate the file, one sentence at a time
5. Clean the file (explained below)
6. Save the file as plain text, selecting the appropriate text encoding

1. Open the file in MS Word

Microsoft Word attempts to guess the file type when opening, and if necessary parses it. If MS Word believes that a file being opened is XML, it will parse it and won't display the tags. When saving such a file, MS Word may recode the file according to its own idea of what XML should look like. For this reason, XML files like XLIFF or TMX can't simply be opened in MS Word, because information will be lost when it is saved.

When a UTF-8 file (or any encoding other than ANSI or ASCII) is opened in MS Word (either by drag-and-drop or by using the File Open menu item), MS Word will guess the encoding and prompt the user to select it from a list.

The trick for our purposes is to force MS Word to prompt the user when opening an XML file. To this, open a blank document in MS Word, then go *Tools → Options → General → Confirm conversion at open*. Then open the file by using the *File → Open* menu item (not by drag-and-drop). If all goes well, MS Word will ask what the file type is (answer "Encoded text") and what the encoding is (usually UTF-8).

MS Word has something called "Unicode" in addition to "UTF-8" – this is a special Microsoft kind of Unicode that is not really compatible with UTF-8.

2. Save the file as RTF or MS Word

Wordfast adds invisible codes to the document while translating, and for this reason the file being translated must be saved as MS Word or RTF format while translating. These codes are word processing codes and are not saved in the text file when an RTF file is resaved as text. With MS Word, what you see is what you get (or rather, what you see is what gets saved, if you save as plaintext).

To save a file in MS Word, press `F12`, and type in the file name, and select the file type. RTF is sufficient, but MS Word is usually used by Wordfast users.

3. Prepare the file for translation (optional)

If the file to be translated is a plain text file, then no further preparation is necessary. However, if the file is an XML file or contains sections that shouldn't be translated, it is necessary to prepare the file by marking the text in such a way that Wordfast will ignore text that shouldn't be translated.

In fact, if a Wordfast user who doesn't know how XML works, receives such a prepared file, he can translate it using Wordfast and deliver a perfectly valid XML file in the target language, because Wordfast will not prompt him to touch any text that shouldn't be touched during the translation process.

Preparing the file involves marking text as translatable and untranslatable, and marking the untranslatable text as either internal or external text. To illustrate, take a look at this piece of HTML:

```
<ul>
<li><b>This is a <i>dog</i>.</b> That is a <a href="lion.html">cat</a>.</li>
</ul>
```

In the above example, the text "This is a dog. That is a cat." should be translated. The tags <ul> and <li> are not inline tags, and there is no reason why the user should have anything to do with them. The tags <b> and <i> are inline formatting tags and the tag <a> is also an inline tag. The translator should not change them, but he will likely have to move them around in the translation.

To prepare the above example text for translation in Wordfast, the tags <ul> and <li> should be marked as external untranslatables, and the tags <b>, <i> and <a> should be marked as internal untranslatables.

Wordfast has special routines for marking HTML, XML and a range of other common formats. It doesn't have a routine for PO or XLIFF, or for Mozilla .properties and .DTD files, but a quick Find/Replace macro is all that is needed to accomplish that (a macro will be given on a different page). A tool such as Tortoise Tagger can also be used.

Marking internal untranslatable text can speed up translation, but if a translator knows which texts not to change, it is not necessary for such text portions to be marked as internal. Marking external text is more crucial.

Here's an example from a Mozilla DTD file:

```
<!ENTITY about "About">
```

In the above example, the text "About" should be translated. The text '<!ENTITY about "' and '">' should be marked as external untranslatable.

Here are two lines from a Mozilla .properties file:

```
prefMessage=Int Pref Value: %d
extensions.videodownloader.description=Download videos from Youtube
```

In both lines, the text before the equal sign (and including the equal sign) should be marked as external untranslatable. The variable "%d" can optionally be marked as internal untranslatable, but that is really not essential. If the above two lines are marked up as described, Wordfast will prompt the translator to translate "Int Pref Value: %d" and "Download videos from Youtube", and nothing else.

4. Translate the file, one sentence at a time

When a translator uses Wordfast, Wordfast segments the text on-the-fly into sentences. If Wordfast misguesses where a sentence begins or ends (eg if an unexpected abbreviation occurs in mid-sentence), the translator can easily fix the faulty segmentation there and then.

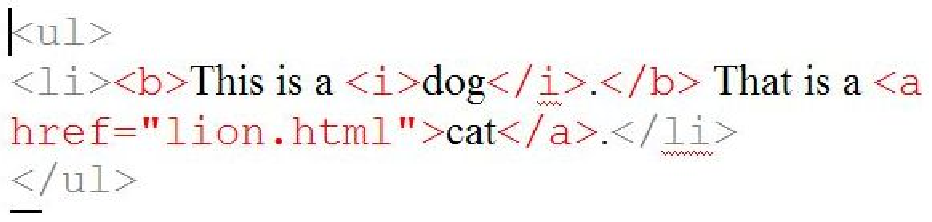
Basically, the translator presses `Alt+Down` to move to the "next segment", then translates it, then presses `Alt+Down` again, and so on.

When the translator presses `Alt+Down`, the next segment is opened in a visual source text box, with an empty target text box below it. The translator then types his translation in the target text box. When the translator presses `Alt+Down` again, the two boxes disappear but both the source and target text remain visible, with some special Wordfast codes between them. These codes are actually hidden text (and the source text is too), so if you press `Ctrl+,`, the hidden text is no longer displayed and you can see what the final text would look like.

Here are some screenshots of the above dog/cat example (external = grey, internal = red, translatable = black).

4.1 Raw text, marked up in styles

The example below is a screenshot taken in MS Word. As you can see (if you know any HTML), the grey text should not be touched by the translator, and the red text should not be changed, although it could be moved around, depending on the language.

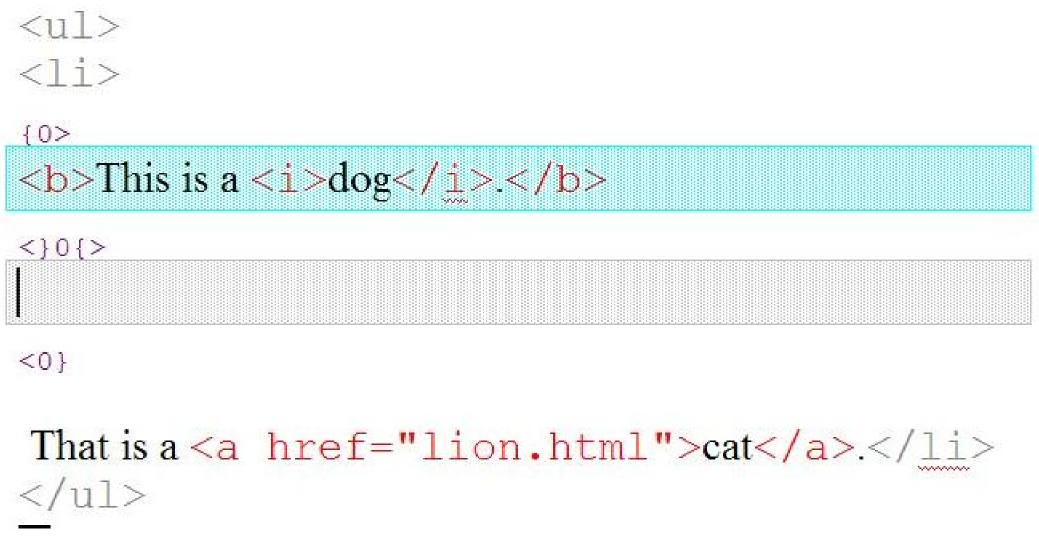


```
<ul>
<li><b>This is a <i>dog</i>.</b> That is a <a
href="lion.html">cat</a>.</li>
</ul>
```

4.2 Alt+Down to start translating

In the screenshot below, the translator had pressed `Alt+Down` (i.e. "next segment") in Wordfast. Wordfast moves the text to be translated to a new line, opens it in a box, and creates an empty box beneath it. The translator will type his translation in the empty box. The purple stuff are codes created by Wordfast to know where a segment begins and ends.

Interestingly, the purple text `<}0{>` is also an indication of whether a fuzzy match for this segment exists in the TM. In this case, there isn't, but if there had been a 77% match, the text would have read `<}77{>`.

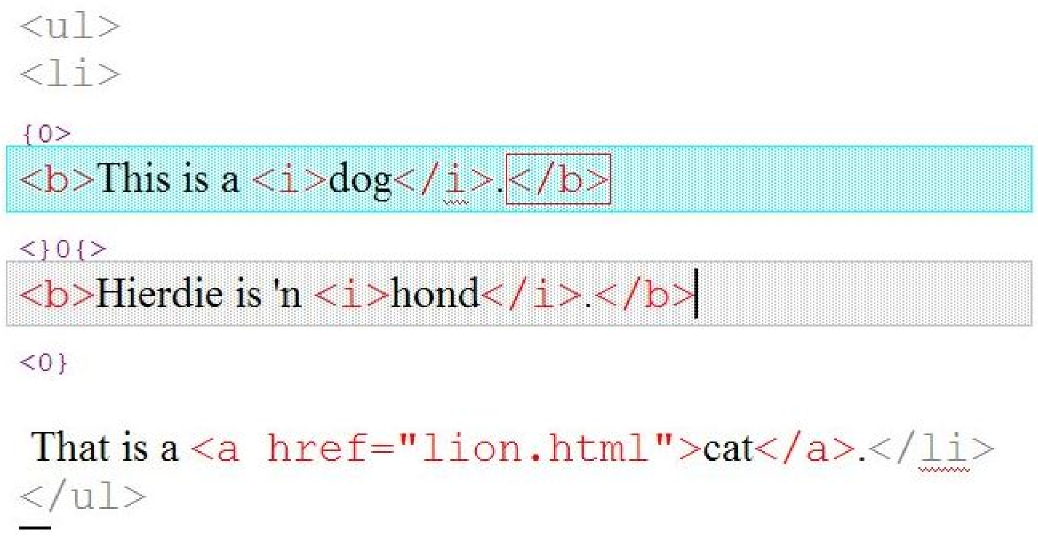


```
<ul>
<li>
{0>
<b>This is a <i>dog</i>.</b>
<}0{>

<0}
 That is a <a href="lion.html">cat</a>.</li>
</ul>
```

4.3 Type in the translation

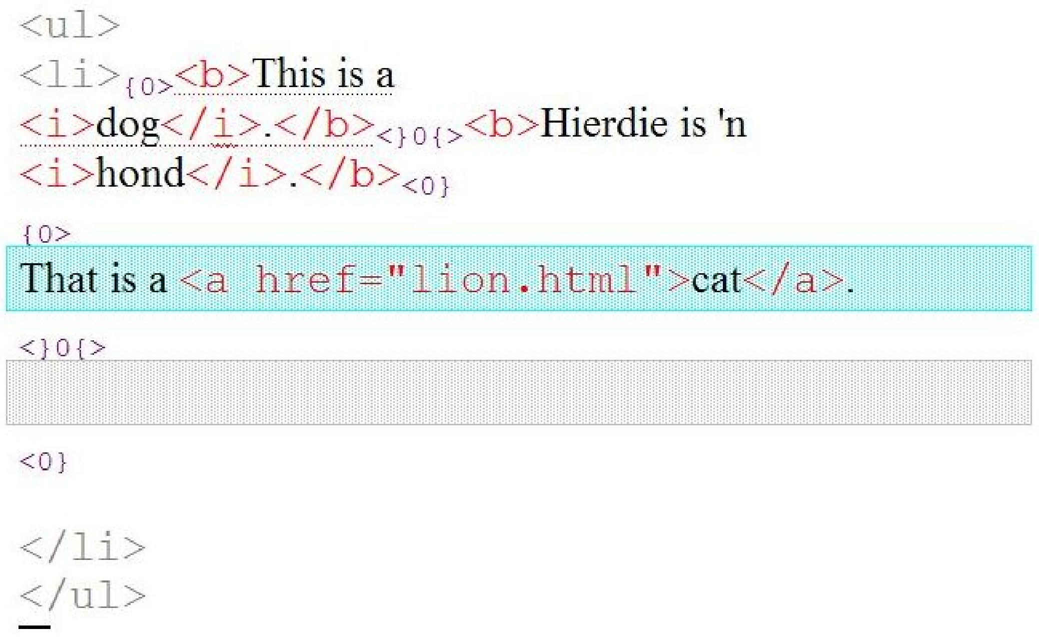
In the screenshot below, the translator had typed in a translation. Strictly speaking, the translator could have typed the red text himself, but Wordfast has a method to grab each piece of internal text from the source box and copy it to the position of the cursor in the target box.



```
<ul>
<li>
{0>
<b>This is a <i>dog</i>.</b>
<}0{>
<b>Hierdie is 'n <i>hond</i>.</b>
<0}
That is a <a href="lion.html">cat</a>.</li>
</ul>
```

4.4 Alt+Down to go to next segment

In the screenshot below, the translator had pressed `Alt+Down` again, to go to the next segment. The previous segment is no longer on its own line, but is inline with the text surrounding it. The purple markers remain, to tell Wordfast where the segment begins and ends.

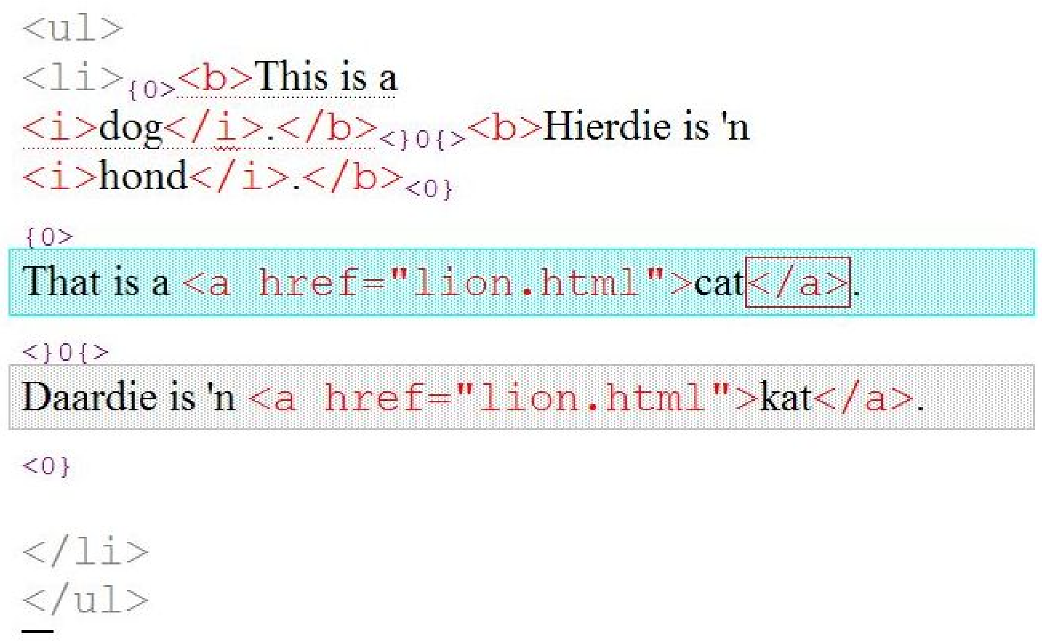


```
<ul>
<li>{0><b>This is a
<i>dog</i>.</b><}0{><b>Hierdie is 'n
<i>hond</i>.</b><0}
{0>
That is a <a href="lion.html">cat</a>.
<}0{>

<0}
</li>
</ul>
```

4.5 Type in the translation again

In the screenshot below, the translation has been typed in again.

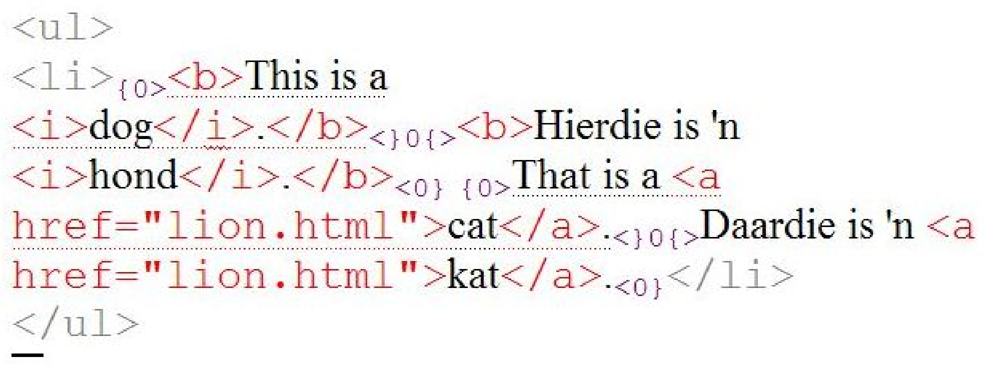


```
<ul>
<li>{0><b>This is a
<i>dog</i>.</b><}0{><b>Hierdie is 'n
<i>hond</i>.</b><0}
{0>
That is a <a href="lion.html">cat</a>.
<}0{>
Daardie is 'n <a href="lion.html">kat</a>.
<0}

</li>
</ul>
```

4.6 End the translation session

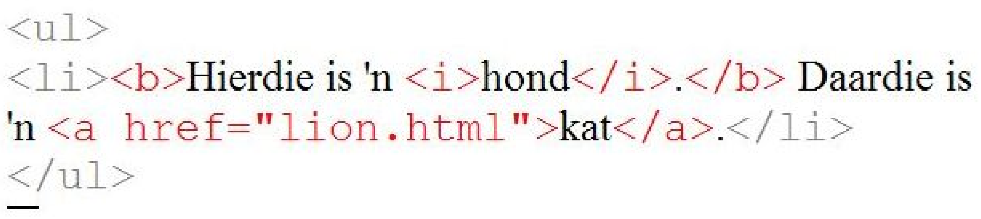
At any time the translator can "end the session" by closing the current segment and not moving on to the next segment. In the screenshot below, the session automatically ended because the last segment was reached.



```
<ul>
<li>{0><b>This is a
<i>dog</i>.</b><}0{><b>Hierdie is 'n
<i>hond</i>.</b><0} {0>That is a <a
href="lion.html">cat</a>.<}0{>Daardie is 'n <a
href="lion.html">kat</a>.<0}</li>
</ul>
```

4.7 Hide hidden text

In the screenshot below, the translator had pressed `Ctrl+,` (i.e. "toggle hidden text"). Only the text that doesn't have the "hidden" attribute is displayed. This is also what the document would look like after clean-up.



```
<ul>
<li><b>Hierdie is 'n <i>hond</i>.</b> Daardie is
'n <a href="lion.html">kat</a>.</li>
</ul>
```

5. Clean the file

A file that has been translated with Wordfast contains both source text and target text, as well as other codes in purple. It may also have text marked as internal or external untranslatable. Even if the client is expecting an MS Word or RTF file back, it is obvious that the translator can't send it back in this form. So the trick is to "clean" the file. The function called "Clean" only removes the purple codes and the source text – it does not remove untranslatable markings.

The screenshot at 4.7 above is what a cleaned file would look like.

The translator can reopen any segment by placing his cursor anywhere in the segment and pressing `Alt+Down` (next segment). He can then edit the translation, and close the segment again. In sucn a case, the TM is updated automatically with the new translation.

The translator could send the "uncleaned" file to colleagues for proofreading. Note that it is possible to edit the uncleaned file without using Wordfast, if care is taken not to overwrite the purple tags. A proofreader can edit an uncleaned file using Track Changes, for example, and the translator or project manager can accept/reject such changes, without affecting the integrity of the uncleaned file. One can also perform a spell-check on the unclean file, and let MS Word correct spelling errors in it.

Changes made to the translations by not opening and closing the segments with Wordfast, will not be reflected in the TM. When a file is cleaned using Wordfast's "clean" function, Wordfast updates the TM by comparing the segments in the document with the segments in the TM.

6. Save the file as plaintext

To use the translation in its l10n context, the file should be saved as plaintext. The translator works on an MS Word document during the translation process, but after he had cleaned the document, he saves it as plaintext. In MS Word, this is called "Encoded text" (the translator gets to choose which encoding).

MS Word will give text files the file extension "TXT". If a different file extension is required, the files should be renamed in MS DOS or using some other renaming tool.

Marking text with styles

Many Wordfast users have never worked with documents that have been marked with untranslatable text. This shouldn't be a problem as long as they are told not to attempt to translate the grey text, and that they should use "Next Placeable", "Previous Placeable" and "Place Placeable" to copy the red text.

Placeables are pieces of text that Wordfast can grab in the source box and copy to the position of the cursor in the target box. Placeables can be placed using icons on the Wordfast toolbar, or using keyboard shortcuts (see the Wordfast manual for a comprehensive list of shortcuts).

Even if a Wordfast user has worked with such marked documents before (they are referred to as tagged texts), he may not know how to mark such a document himself. Ideally, therefore, the marking up of a document will be done by a project manager or senior translator.

Preparing files for Wordfast

It is possible to translate any plaintext l10n file in Wordfast (or Trados, if you know what you're doing). Here's how:

1. Open the file in MS Word, and save as DOC format
2. Prepare it
3. Translate it, check it, proofread it, review it, etc.
4. Clean it
5. Save it as a plaintext file again

If you're a Wordfast user, you will already know how to do steps 3 and 4. Translating a l10n file in Wordfast is no different from translating any other file, except that Wordfast will automatically prevent you from translating certain stuff, and certain pieces of text have to inserted using Wordfast's "placeable" feature (which is quite simple, actually, and is described in the Wordfast user manual)

How to open a l10n file in MS Word

When opening a text file in MS Word, MS Word will try to interpret the file. This is not good, for our purposes – we want MS Word to treat plain text as plain text. If you open an XLIFF file in MS Word, MS Word will recognise it as an XML file, and will parse it as an XML file, rendering the file useful for a passive user, but useless for a translator.

To prevent MS Word from parsing the file, do the following (MS Word 2000 instructions):

1. Open a blank document in MS Word
2. Go to *Tools* → *Options* → *General*
3. Select Confirm conversion at open
4. Click *OK*, etc
5. Close the blank document

MS Word doesn't seem to remember this setting, so you have to check that it is enabled every time you open a l10n file.

Normally there are many ways to open a document in MS Word, eg drag-and-drop, `Ctrl+O`, *File* → *Open*, and double-clicking. For MS Word to respect the setting mentioned above, the l10n file has to be opend using the *File* → *Open* method.

Usually, when opening a l10n file, MS Word will ask you to confirm the type of file. Do not choose "HTML", for example. Choose "Encoded Text", and when prompted for the encoding, select the encoding applicable to the file. Do not choose the option named "Unicode" – be more specific than that. If your file is in UTF-8, choose UTF-8 as the file open type.

Once the file is open in MS Word, save it as a DOC file by pressing `F12` and selecting the DOC option.

At the end of the translation process, if you're certain that no hidden codes are left in the file, you can save it as plain text again, by pressing `F12` and selecting "Encoded Text" as the file save type, and choosing the correct encoding (again, do not choose simply "Unicode" but be more specific). MS Word might complain that you will loose formatting, but hey, that's exactly what you want.

How Wordfast knows which text is which

To understand how Wordfast knows which text should be translated and which shouldn't, one has to understand the concept of styles.

Styles in word processing

A word processor like MS Word (and OpenOffice.org) uses a concept called "styles" to simplify document formatting, although very few people use it. You can apply a style to any piece of text, and that text will look and act according to how the style defined.

You can create a style named "text123" and define it as "red, bold, Times New Roman". If you then select any text and choose the style "text123", the text will become red, bold and Times New Roman. In addition, the text will carry the hidden label "text123".

Once text is marked with a certain style name, you can do all sorts of things with it. You can tell MS Word to delete all "text123" text, and it will delete only the text marked in that style, even if there are other pieces of text that look exactly the same (red, bold, Times New Roman). You can also change the style's definition to, say, "red, italics, Times New Roman", and all text marked in the "text123" style will automatically become italic and non-bold.

For Wordfast, what a style looks like, is irrelevant, but the name of the style is important. Wordfast specifically looks for two styles, called tw4winExternal and tw4winInternal. The former is usually grey, the latter is usually red.

How to create Wordfast styles (the easy way)

The easiest way to create Wordfast styles, is to open a document that has Wordfast styles in it, and then add those styles to your computer's normal.dot generic template. If you do this, your MS Word's Wordfast styles will look like those of everyone else in the world (grey or red, hidden, Courier New, etc).

Here's how:

1. Open the document that contains Wordfast styles
2. In MS Word, go *Format* → *Style*, and click *Organizer*
3. In the organiser you'll see two documents open, namely the normal.dot and your current document. Simply select the tw4win styles in the current document, and click *Copy* to copy them to the normal.dot.
4. Close both documents in the organiser, and click *OK* etc
5. Close MS Word, and start MS Word again. See if the tw4win styles are now present in the styles dropdown list (usually upper left corner).

Note: You can download a prepared tw4winstyles.doc document.

How to create Wordfast styles (easy way, not ideal though)

You can create the requires styles in MS Word yourself, manually, but you may find that the styles do not behave exactly the way that standard Wordfast styles behave. For example, if you forget to specify "No Proofing" as an attribute, you may find that MS Word tries to spellcheck the raw l10n code.

To do this manually, you need to know what the exact style names are, because really that's all Wordfast really cares about. Here's how:

1. Open a blank document in MS Word, or open any other document in MS Word
2. In MS Word, go *Format* → *Style*, and click *New*
3. Name the style tw4winExternal, and make it a "Character" style
4. Click the *Format* button, and select various pieces of formatting
5. Click *OK*, etc.

The style is now part of the current document. To have the style available for other documents, you should add it to the normal.dot template, described above in "the easy way". You may also create the style very time you open a new l10n file, if you enjoy doing that.

We suggest the following formatting for the tw4win styles:

tw4winExternal

- Font: Courier New, Regular, 12, Grey
- Language: No Proofing

tw4winInternal

- Font: Courier New, Regular, 12, Red
- Language: No Proofing

How to create Wordfast styles (if you can install external macros)

If you know how to install external macros (i.e. if you know where you should copy a file in MS Windows' hidden folder structure), you can install AndoTools into MS Word, which has a function to insert all tw4win styles into any document easily. Once you've installed AndoTools, in MS Word go *Undo* → *Document Operations* → *Fonts and Language*. Click "Add tw4win styles" to add them to the current document.

How to prepare a l10n file for Wordfast

The concept of preparing a l10n file for Wordfast, is actually quite simple. All you need to do, is to mark text that shouldn't be translated, as tw4winExternal, and possibly any text that may be moved around, as tw4winInternal. What's more, the tw4winInternal is really only for advanced, complex stuff like certain types of XML. And even if a document can use tw4winInternal, not having it will not make a difference as long as the translator knows which pieces of text he should and shouldn't change.

For example, in the following line:

```
| The <bold>quick</bold> brown fox... |
```

the translator should know that <bold> and </bold> should not be translated, but kept in "English". These two pieces of text can be marked as tw4winInternal, to help a translator copy them easier, but it isn't absolutely necessary.

Marking tw4winInternal is a lot more work than marking tw4winExternal, so don't bother, to begin with.

Preparing a file manually (the hard way)

I'm going to show how to prepare a file the hard way because it offers a useful introduction to MS Word's advanced find/replace functions. MS Word can do limited regular expressions, with certain types of backreferences, which can be quite useful.

What we're going to do, is to mark a document with tw4winExternal. It is assumed that either normal.dot or the document will itself have a style called tw4winExternal already defined. The easiest document to practice on, is a Mozilla DTD file called about.dtd. Open the file `about.dtd` in MS Word as describe above. The encoding is UTF-8.

The file looks like this:

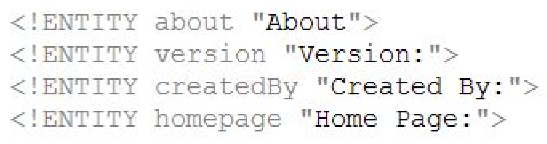
```
<!ENTITY about "About">
<!ENTITY version "Version:">
<!ENTITY createdBy "Created By:">
<!ENTITY homepage "Home Page:">
```

The stuff that needs translating, is between quotes. The quotes themselves should not be translated – they do not form part of the "translatable" text. Therefore, we must mark everything from `<! ENTITY` to `"` as tw4winExternal, and everything that is `">` should also be marked as tw4winExternal.

Here's how we do it:

- In MS Word, press `Ctrl+H` (the find/replace box). Click "More" to open advanced features.
- Placing your cursor in the Find box, type `(\<\!ENTITY)(*)(\")`.
- Place your cursor in the Replace box, and type `\1\2\3`.
- Make sure your cursor is still in the Replace box, then click *Format* → *Styles*, and select tw4winExternal from the list
- Select "Use Wildcards", and click "Replace all"
- Then, place your cursor in the Find box, and type `">`.
- In the Replace box, remove everything (it must be empty). Check that the style at the Replace box is tw4winExternal (if not, add it)
- Deselect "Use Wildcards", and click "Replace all"

The result should look like this or like this:



```
<!ENTITY about "About">
<!ENTITY version "Version:">
<!ENTITY createdBy "Created By:">
<!ENTITY homepage "Home Page:">
```

This DOC file can now be sent to a Wordfast user, who can translate it without having to worry about which texts he should touch and which not, because Wordfast will only prompt him to translate the black text.

Preparing a file manually (the easy way)

The DTD file above had a very simple structure, and it was simple to tag using find/replace. However, some formats are more complex, requiring many, many steps of finding and replacing. Luckily, MS Word allows us to record a number of steps and save it as a macro. The ideal would therefore be to create a macro for each type of l10n file, and simply use the macro.

In MS Word, a macro can be embedded in a document so that it can be transported and included into another document (or ideally in the project manager's normal.dot template).

Adding a macro by pasting it

Let's add the following macro to MS Word's normal.dot.

```
Selection.HomeKey Unit:=wdStory
Selection.Find.ClearFormatting
Selection.Find.Replacement.ClearFormatting
With Selection.Find
    .Text = ""
    .Replacement.Text = ""
    .Forward = True
    .Wrap = wdFindContinue
    .Format = False
    .MatchCase = False
    .MatchWholeWord = False
    .MatchWildcards = False
    .MatchSoundsLike = False
    .MatchAllWordForms = False
End With
Selection.Find.ClearFormatting
Selection.Find.Replacement.ClearFormatting
Selection.Find.Replacement.Style = ActiveDocument.Styles("tw4winExternal")
With Selection.Find
    .Text = "(\<\!ENTITY)(*)(\"")"
    .Replacement.Text = "\1\2\3"
    .Forward = True
    .Wrap = wdFindContinue
    .Format = True
    .MatchCase = False
    .MatchWholeWord = False
    .MatchAllWordForms = False
    .MatchSoundsLike = False
    .MatchWildcards = True
End With
Selection.Find.Execute Replace:=wdReplaceAll
Selection.Find.ClearFormatting
Selection.Find.Replacement.ClearFormatting
Selection.Find.Replacement.Style = ActiveDocument.Styles("tw4winExternal")
With Selection.Find
    .Text = """>"
    .Replacement.Text = ""
    .Forward = True
    .Wrap = wdFindContinue
    .Format = True
    .MatchCase = False
    .MatchWholeWord = False
    .MatchWildcards = False
    .MatchSoundsLike = False
    .MatchAllWordForms = False
End With
Selection.Find.Execute Replace:=wdReplaceAll
```

This macro was recorded, and I'm sure any Visual Basic programmer could trim it down to less lines.

To add the above macro, do the following:

- Select and copy the macro (copy to clipboard, `Ctrl+C`)
- Open a blank document in MS Word.
- In MS Word, go *Tools* → *Macro* → *Macros*.
- Type in the macro name, say, "apple" (use a name at the beginning of the alphabet, to find it easily).

- Click "Create".
- Place your cursor in the line above "EndSub" (by default your cursor will be there).
- Paste the above macro at that point.
- Press `Ctrl+S` to save, and exit the macro writer

The macro is now added to normal.dot, and can be used for any document that is opened in MS Word. Incidentally, the above macro does exactly what we did in the advanced find/replace operation above.

Adding a macro from an existing document

Adding a macro to normal.dot from an existing document is similar to what we did in the "easy way" for adding styles. I assume you have a document with a macro embedded in it. I've embedded the above macro for you, in a document.

To add the macro to normal.dot, here's how:

- Open a blank document in MS Word.
- In MS Word, go *Tools* → *Macro* → *Macros*.
- Click *Organizer*. It should show you the macros in with_apple.doc and normal.dot.
- Select "apple" and click *Copy* to copy to normal.dot.
- Close both files, and click *OK* etc.

And that's it. Now a macro called apple.apple is part of normal.dot, and can be used on any document you open in MS Word.

How to execute a macro on a document

When running the macros described above, it is assumed that you have tw4winExternal as a style in normal.dot, or in the document that you're about to tag. What we're going to do, is to run the macro apple or apple.apple, which will perform the find/replace operation mentioned previously. This will mark the necessary text as "untranslatable", so that Wordfast will ignore it.

- Open the l10n file in MS Word (described above)
- In MS Word, go *Tools* → *Macro* → *Macros*.
- Select the macro apple or apple.apple in the list, and click "Run" (if you can't see the macro, it is either not in the normal.dot, or the normal.dot is not selected in the dropdown list).
- Click OK etc.

If everything went well, your document should now be tagged, as per the images above.

(next write a short intro, plus upload a number of macros for XLIFF, TMX, PO, etc.

Another way…

Some of what is written here, is re-inventing the wheel. The wheel we're talking about, is Tortoise Tagger.

Creating a Translation Memory with WordFast

Dwayne Bailey wrote on 24/01/2006 09:16 PM:

> On Tue, 2006-01-24 at 15:33 +0200, Samuel Murray wrote:

>> Here's what I would have done (if it were just me)…

> All this download, freeware, right click stuff makes me nervous :)

Hey… I thought Linux people thrived on modular approaches!

>> What I would have done, was to create a translation memory using the
>> translated strings of 1.0.5, …

> You don't say how.

Okay, first step, try to think like a Windows user. Then… well,
okay… just kidding, skip that step :-)

1. Download firefox-af-1.0.7.tar.gz and unzip it.
2. Open all the PO files in MS Word (well, ten at a time is better),
and save them in MS Word or RTF format. Tip: open the files by dragging
and dropping into an empty MS Word instance.
3. Convert the PO files into memories (there are two ways of doing
this, Method A and Method B, each with pros and cons).

But first: What is an uncleaned Trados file?

Right… how to explain this… erm… see, when you translate a file
using Wordfast (or Trados), the macros in Wordfast split the text into
segments, and the translator translates one segment at a time. When the
translation is finished, the end-result is an MS Word or RTF file known
as an uncleaned document (or, an uncleaned Trados file). The file looks
like this:

{0>This is a house.<}0{>Dit is 'n huis.<0} {0>This is another
house.<}77{>Dit is nog 'n huis.<0} {0>This is not a house.<}85{>Dit is
nie 'n huis nie.<0} {0>These are not houses.<}35{>Hierdie is nie huise
nie.<0}

The numbers indicate the fuzzy match percentage calculated during the
translation process. Don't worry about the numbers… under different
circumstances they have different meanings, so forget about them for now.

The source text is usually marked as "hidden text" in MS Word, but this
is not required. The tags {0>, <}[0-9][0-9]{> and <0} are in a unique

style known as tw4winMark (usually displayed as purple text).

The client might give this uncleaned file to a proofreader to check the
translation. The proofreader might make several changes to the target
pieces of text. When the client receives the proofread text back, he
would naturally want the changes to be updated in his translation
memory, right? So he loads the proofread uncleaned document into
Wordfast, and he loads his original translation memory (which was
created when the document was originally translated) into Wordfast, and
he performs a "cleanup". The cleanup removes the source text, it
removes the tags, and it checks every segment against the translation
memory, and updates the translation memory when it finds a changed segment.

And now for the nice one: If during the cleanup, Wordfast finds a
segment in the uncleaned document which is not in the translation
memory, it adds that segment to the translation memory. It follows,
therefore, that if you do a cleanup using an *empty* translation memory,
that all the segments will be added to the translation memory during
cleanup. And this is the key to Method A mentioned above.

Method A

Convert this:

# This is a comment.
# This is another comment.
msgstr "I don't use Linux if I can help it."
msgid "Ek gebruik nie Linux as ek dit kan verhelp nie."

into this:

{0>I don't use Linux if I can help it.<}0{>Ek gebruik nie Linux as ek
dit kan verhelp nie.<0}

and then do a cleanup on an empty translation memory.

Pros: quick and easy
Cons: segmentation according to PO rules, not Wordfast rules

Let me explain the segmentation issue here. According to this method,
this string:

# This is a comment.
# This is another comment.
msgstr "I'm a very happy Windows user. I'm a very happy Windows user.
I'm a very happy… I'm a very happy… Oh, I'm a very happy Windows user."
msgid "Ek's 'n baie bly Windows-gebruiker. Ek's 'n baie bly

Windows-gebruiker. Ek's 'n baie bly… Ek's 'n baie bly… O, ek's 'n
baie bly Windows-gebruiker."

will be turned into this:

{0>I'm a very happy Windows user. I'm a very happy Windows user. I'm a
very happy… I'm a very happy… Oh, I'm a very happy Windows
user.<}0{>Ek's 'n baie bly Windows-gebruiker. Ek's 'n baie bly
Windows-gebruiker. Ek's 'n baie bly… Ek's 'n baie bly… O, ek's 'n
baie bly Windows-gebruiker.<0}

which means that there will be one segment containing five sentences.

According to Wordfast segmentation rules, however, there should be at
least three segments:

{0>I'm a very happy Windows user.<}0{>Ek's 'n baie bly
Windows-gebruiker.<0} {0>I'm a very happy Windows user.<}100{>Ek's 'n
baie bly Windows-gebruiker.<0} {0>I'm a very happy…<}0{>Ek's 'n baie
bly…<0} {0>I'm a very happy…<}100{>Ek's 'n baie bly…<0} {0>Oh, I'm
a very happy Windows user.<}94{>O, ek's 'n baie bly Windows-gebruiker.<0}

This means that PO strings containing more than one Wordfast segment
will not get any matches (or will get less useful matches) from a
Wordfast translation memory.

Method B

Method B is to create two files – one containing only the source text
and one containing only the target text – and align them. Method two
takes a bit longer, and involves manual labour, but it produces a
translation memory which is very, very useful because it uses exactly
the same segmentation rules as those which would be used during the
final translation.

Ideally, one would use both method A and B, and then combine the two
memories into a single super memory, which would yield more matches.

Since the Firefox old translation is not a single file but several
files, it would be easier to use Method B on the memory which was
created using Method A, than to attempt to extract the source and target
text from the 100 PO files and hope to maintain the sentences in an
alignable order.

Wordfast has an "extract" feature in which it segments the text of a
document into segments, each on its own line. Hence, if you run an
extract on this paragraph:

This is a house. This is another house. This is not a house. This is
the best house I have ever seen. This is a house. These are houses.

the end-result would be:

This is a house.
This is another house.
This is not a house.
This is the best house I have ever seen.
This is a house.
These are houses.

and if you have a translation of this document, the translated strings
could also be extracted to something like:

Dit is 'n huis.
Dit is nog 'n huis.
Dit is nie 'n huis nie.
Dit is die beste huis wat ek al gesien het.
Dit is 'n huis.
Hierdie is huise.

and then, because the source and target sentences in these two extracted
documents are mostly in the same order, you can align these two
documents in table form (or another form) and manually check to see if
each source segment fits the target segment that follows it.

Then correct the misalignments, press a button (Create TM), and let
Wordfast create a translation memory based on these segments. Et voila.

Does this answer your question? Any other questions?

Samuel

Translating PO files with WordFast

I'd like to make some notes that can be reused for someone using
WordFast if they wanted to translate PO files.

Well, the important thing to remember is that Wordfast is not intelligent. The intelligence lies with the translator and/or the guy writing the tagger. Thankfully PO is quite simple, and if the lines are not wrapped, a translator can tag it almost manually.

What is tagging?

Well, in an MS Word document, you can apply a "style" to certain pieces of text. Default styles include Heading 1, Paragraph, etc. You can also set the behaviour of a style, so that all text with that style will look the same way. For example, I could set "Heading 1" to be Arial font, size 12, red text on a blue background, and all text with the style Heading 1 will look that way. It's kinda like HTML. You can also create your own "styles", and make them look like any thing you want to. The nice thing about styles is that you can tell a macro or a Find/Replace operation to perform its actions only on certain styles instead of the entire text. Very few people who use MS Word are aware of the great possibilities that exist thanks to "styles".

If I were to translate an HTML document in MS Word, I could do it as follows: First copy the text (the HTML code) to an empty MS Word document. Then create a unique style with a unique name. Then use Find/Replace to change the style of all the non-translatable code to that unique style. Then write a macro which extracts the translatable text by ignoring everything which is in that unique style. Or, write a macro which hides all text which is in that unique style (so that only the translatable text is visible). Or, well, you can see the possibilities here.

In Wordfast (and in Trados) there are two such unique styles, with which the macros interact in a special way. The one style is called tw4winExternal and the other is called tw4winInternal. In an HTML file (saved as an MS Word document), the External style is applied to block-level code and the Internal style is applied to line-level code. The Wordfast macro ignores everything which has the External style, and creates a temporary text box with each sentence of translatable text in which the translator types the translation. Text with the Internal style has a red colour, so that the translator can see which pieces of line-level HTML he should watch out for. In this way, the translator doesn't need to know HTML and yet he can translate directly within an HTML document (using MS Word) without any fear of "breaking" the code.

Marking the untranslatable code with External and Internal styles, is known as "tagging".

So, a translator faced with a tagged PO file in MS Word will not necessarily know how the PO format works. He will be able to see everything in the PO file, but he'll ignore it because Wordfast will automatically allow him access only to the text to the translated.

And now for the complication…

In Wordfast, the source text is *overwritten* with the target text. This means that when the translator gets the file from the client, the file must already be in such a way that the source text can be overwritten by the target text. This is why I insist on a "duplicated" PO file (however, I can duplicate it in MS Word too, so if the client is unable to provide a "duplicated" file, I can do it myself anyway).

Now, if you think about it carefully, you'll see that a translator *cannot* deal with PO fuzzys in Wordfast. Not possible.

However, most of the time when fuzzys occur, it is because an existing string (which had been translated before) has changed slightly, right? So it follows that the old string and its correct translation is still available somewhere. If this old string and its translation isn't in the translation memory (TM) already, it can easily be added to it, right?

Therefore, a simple workmethod is to delete all the proposed translations of fuzzys and turn the fuzzys into untranslateds (and then "duplicate" them), and translate them as usual.

Translating Opera

When I get PO files from my contact at Opera, the translated, fuzzy and untranslated strings are in a single file. First I used Gnu Gettext tools to separate these into three files. Then I turn the fuzzy strings into untranslated strings. Then I do the translation on the two "untranslated" files, and then I merge them back into the original file (the translated strings remain untouched by this process).

With Dwayne's PO files, I have to be more careful, and that is why I prefer that he does the gymnastics (then if something goes wrong and it is a conversion issue, then it's his fault). The gymnastics I'm referring to here, is separating the translated from the untranslated, and putting them into separate files. The "duplication" I can do flawlessless in MS Word (as long as the lines are unwrapped).

What is duplication (just in case you don't know)?

Here is an unduplicated PO string:

```
# This is a comment.
# This is another comment.
msgstr "This is the source text."
msgid ""
```

Here is a duplicated PO string:

```
# This is a comment.
# This is another comment.
msgstr "This is the source text."
msgid "This is the source text."
```

Here is a translated PO string:

```
# This is a comment.
# This is another comment.
msgstr "This is the source text."
msgid "Hierdie is die bronteks."
```

These examples are oversimplified, of course.

Any questions?

The reason why some of the first PO files I did for Dwayne, didn't quite work out well, was because I didn't know how to do regex back then, so I didn't know how to do tagging, so instead I did a roundtripping which turned out to be only 99% lossless. Ouch (and again: sorry).

Samuel

PrepTags

PrepTags is a file preparation software which allows to prepare files for translation in most CAT tools. Export formats include tagged RTF (Wordfast, DV, Trados, MemoQ), ttx (native TagEditor format), XLIFF (Virtaal, OmegaT, Heartsome…), TXML (native format for Wordfast Pro) and text (simplified tagged RTF format.

Files are prepared by defining a taglist, which is essentially a list of regular expressions enhanced with tagging features. This allows for the preparation of almost any text-based format, including XML, php, asp, po files…, even if they do not follow the standard rules of the format.

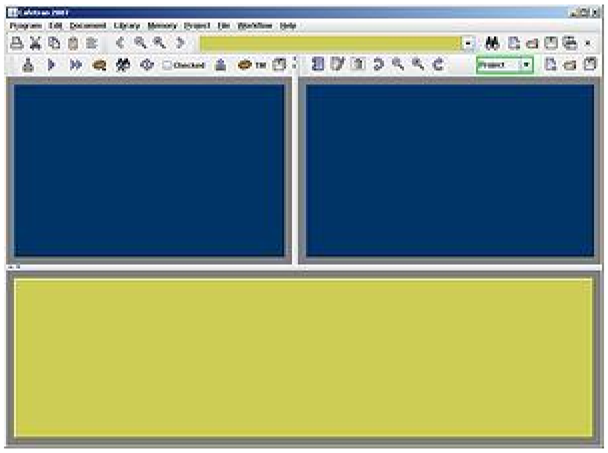
http://www.preptags.com

PrepTags is not open source, but there is a free, functional Lite version.

Cafetran CAT tool

Cafetran is a CAT tool with translation memory, glossary lookup, and web searching. It also supports sourceless translation using an autocomplete function.

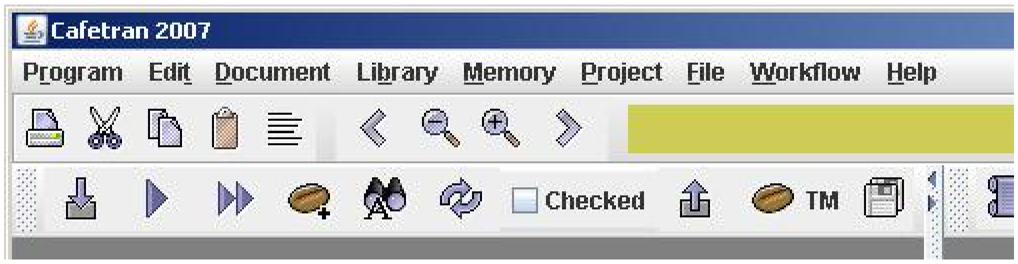
https://www.cafetran.com



Cafetran is written in Java. Cafetran is not free, but a time-limited trial version can be downloaded from its web site.

What makes Cafetran interesting is that it uses XLIFF for its native TM format. Documents translated in Cafetran automatically generate an XLIFF file as you go along.

Cafetran is fairly complicated to use, and the menu bar is very non-standard. The help file contained on the web site is not very comprehensive or well organised.



Source files include HTML, OpenDocument, WordML (Microsoft Word 2007), etc.

Open Language Tools

XLIFF editor

https://open-language-tools.dev.java.net/

- Overview of *Common CAT tools*, and how they handle l10n files files
- Wordfast
  - *Introduction to Wordfast*
  - *Preparing files for Wordfast*
  - *Creating a Translation Memory with WordFast*
  - *Translating PO files with WordFast*
- *PrepTags* http://www.preptags.com
- *Cafetran CAT tool*
- *Open Language Tools*

1.14 Fonts, characters and rendering

1.14.1 Unicode

Unicode is a standardised character encoding that allows all characters in all languages of the world to be represented in one character set. It makes it much easier to work with characters and to allow different characters into the same document eg Chinese, Arabic and Roman.

Useful resources

The following are a list of URLs for useful information on Unicode:

- http://www.unicode.org/ – the Unicode home page
- Richard Ishida's
  - Unicode character picker – write the characters of various languages
  - UniView – lookup details and see all Unicode characters
- http://www.eki.ee/letter/ – an online database that will show you what the characters look like, what characters you need for you language and their various coding in various other character sets. This is a very useful site.
- List of various Unicode utilities

1.14.2 Free and Open Source Fonts

If your language doesn't use traditional Latin characters then most likely you will need to look at fonts. Here we try to list fonts that you can actually change to add your characters.

Resources

- Optimal Use of Fonts on Linux
- Ed Trager's Unicode Font Guide For Free/Libre Open Source Operating Systems
- The Wine projects discussion on creating missing Windows fonts
- Scribus page on fonts
- FontForge – Free font editing and creation tool
- Indrek Hein's Letter Database – what special characters are needed by your/a language, their Unicode codes, etc
- Fontconfig Orthographies – these fc-lang files contain the list of characters needed by various languages and are used to better guess which font to use to render a document.
- http://www.travelphrases.info/fonts.html – want to see the scripts of various languages and their fonts, plus links to both free and proprietary fonts.
- Troubleshooting
  - Unix Fonts and AbiWord
  - OpenOffice.org's Font Trouble Shooting Guide
- Freedesktop.org's list of:
  - Open Source and other fonts

  - BSD and GPLd fonts – this covers many non-Latin fonts
- A survey of free font licenses (NewsForge article)

Font List

We probably need to start classifiying this list: decrotive, bitmap, etc.

- Bitsream Vera (2) font – you can't change this one see DejaVu if you want to add your languages characters
- DejaVu – if you have to extend Bitstream Vera then do it in this font.
- Greenville – Tahoma replacement (comparison) – seems to be vapourware
- URW – the 35 standard PostScript(TM) fonts, donated under the GPL by URW++ Design and Development GmbH.
- X.org's list of Open Source fonts
- Free UCS Outline Fonts – This project aims to provide a set of free outline (PostScript Type0, TrueType, OpenType. . .) fonts covering the ISO 10646/Unicode UCS (Universal Character Set).
- Computer Modern Unicode Fonts
- UCS bitmap fonts
- Wine Projects fonts replacements for Windows courier, marlett, sans serif and system
- MgOpen fonts cover Greek but also Latin characters and are licensed in the same way as Bitstream Vera ie you can create derivatives if you use another name. They also mention that you can try to get your glyphs accepted into the main font which seems like a good option, otherwise you would need to start an effort similar to DejaVu
- Matt Chisholm's free decorative fonts (GPLd)
- Dark Garden decorative font
- Artwiz bitmap fonts
- Libertine fonts. Regular, italic, bold and small caps. Regular has very good latin coverage the others are still in progress.
- John Stracke's decorative fonts
- Some bitmap fonts on the Yudit site.
- Junicode a font for medievalists. This has pretty good general latin coverage.

Determining Font Coverage

What characters are included in a font? This is often hard to work out and tedious if you have to type and check characters. Its easier to let fontforge do the work for you. Here are the steps to be followed:

1. Open the font file(s): TTF, etc using fontforge
2. Save as a fontforge .sfd file
3. Use status.pl from the DejaVu project to create a coverage file

This example creates a coverage file for Micrsoft's Arial Unicode font:

```
$ fontforge ARIALUNI.TTF # File Save As... and make sure you save an SFD file
$ touch ArialUnicodeMS.status.txt
$ ./status.pl original ArialUnicodeMS.status.txt ArialUnicodeMS.sfd  >>␣
↪ArialUnicodeMS.status.txt.new
$ vim ArialUnicodeMS.status.txt.new # to view the coverage
```

Coverage – a future hack

Ideally you need a small app that can output coverage or more ideally if fed a font or font directory plus a list of required code points it will return a list of fonts that can satisfy the requirements. It should run on either Windows or Linux, be able to find the system font directory by default or be supplied with a directory or font file. It must be able to output all code points covered or return a coverage percentage if a list of required code points is supplied.

The following links could prove useful in building such an app:

- MSDN GetFontUnicodeRanges, GLYPHSET and WCRANGE
- A FreeType test that seems to test coverage: http://www.scipy.org/mailinglists/mailman?fn=scipy-cvs/2003-November/002280.html
- FontTools
- TTFQuery

1.14.3 Microsoft's Core Fonts

An unfortunate legacy of the browser wars is that many sites are designed specifically for Microsofts Internet Explorer and more importantly are designed around a few Microsoft specific fonts. Fortunately Microsoft released these fonts gratis on the web. Note however, that the Tahoma font does require a valid Windows License.

Installation instructions

The corefonts project has links to the required files and instructions for building and installing the fonts on Linux.

Windows users without these fonts can simply download the required files.

1.14.4 Other fonts

If you cant't get what you need from the *Free and Open Source Fonts* then here is where you can look.

License unknown

- STIX fonts – a project to create a complete set of fonts for technical publication. They are nearing the completion of their goal. They are still to determine a usage license

1.14.5 Keyboards

For some languages you will need to create a keyboard. For many Latin languages this is not that difficult for others it can get quite elaborate and time consuming. Its easy if you are simply capturing the keyboard operation from an existing platform eg migrating a Microsoft based keyboard to Linux. However, if you are creating a whole new keyboard then you have a number of issues such as acceptance of the keyboard and standardisation processes.

Resources

- SIL documents on keyboards and input methods
  - Guidelines for Writing System Support: Technical Details: Data Entry and Editing
  - An introduction to keyboard design theory: What goes where?
  - SIL Keyboarding Chart for Africa
  - Resources for Writing Systems Implementation using Copyleft and FLOSS (Free/Libre and Open Source Software)
  - Some tools and resources for character input
- A list of Alternate Input Methods for Windows

Creating keyboards on Microsoft

Microsoft Keyboard Layout Creator (MSKLC)

- MSKLC – Microsoft Keyboard Layout Creator

This will only work on Windows 2000 and XP. It WILL NOT work on Windows 95, 98, ME or NT 4

Using MSKLC

Firstly load an existing keyboard layout if you simply want to extend and existing layout.

You will notice that you can define keys for for each of the shift states and for the other modified state (usually `AltGr`). For dead keys you must define one of the states as being a dead key and then define each of the combinations. Quite easy but tedious.

Once you are happy with your keyboard then select *Project → Test Keyboard Layout...* you will see a text entry field and can type and test the keyboard. If all works well then select *Project → Validate Layout*, this will give you a report about any problems that were detected in your keyboard.

Want a picture of your new keyboard? *File → Save As Image* – they're ugly but they get the message across.

The last step is to build your keyboard for that select *Project → Build DLL and Setup Package*. This will create and MSI installer and a keyboard DLL, see the section below for steps on how to actually make the installer work.

Repackaging Installers with WiX

The MSKLC creates .msi files that don't contain the dll (they put it in a separate file).

In the keyboards/za module in CVS there is an example of how to rebuild these files correctly (for the South African keyboards). You need to run Make xx-xx.msi (and have the WiX binaries installed, see below) to rebuild the msi file for a language. You will probably need cygwin to run Make (it requires sed as well).

Manual way to fix this:

1. download WiX binaries from http://sourceforge.net/projects/wix/ and unpack them somewhere.
2. Disassemble the .msi file: `/path/to/wix/dark.exe -x . path/to/us-za.msi us-za.wxs`
3. Make the changes to the .wxs file listed below
4. Make the wix object file: `/path/to/wix/candle.exe us-za.wxs`

5. Build the new installer: `/path/to/wix/light.exe us-za.wixobj`

You should see that the built installer has increased in size.

Changes need to the MSKLC to get it to make a standalone installer

Edit the .wxs file before the candle and light commands:

1. Add **Compressed="yes"** as an attribute to the Package element at the start
2. Add **EmbedCab="yes" Cabinet="Keyboard.cab"** to the Media element a bit further down
3. Move the Media entry to between the **Package** and **Property** sections.

Using MSKLC automatically

It's nasty using a GUI tool :-)

http://lists.sharif.edu/pipermail/persiancomputing/2004-June/001425.html seems to be an approach to making it command-line…

Janko's Keyboard Generator

- Janko's Home Page

This will only work for Windows 95, 98 and ME.

Please note that if your languages characters are not in any Microsoft codepages then you will not be able to create a keyboard with those characters. Michael Kaplan (the Microsoft localisation guru) explains why not in this blog entry (look at the second one). Essentially Windows 95 is an old product which MS has stopped supporting and they'd rather you move to something new, they're both technical and marketing considerations.

Creating keyboard layouts for Linux

X.org and XFree86

FIXME various pointers

Getting a nice picture of your layout

This proves rather harder than expected. These pages will give you some pointers that are hopefully helpful.

- Ogonkify patch for processing xkbprint output
- Look at xkbprint man page

Keyboard Mapping for Linux (KMFL)

- Sourceforge project page

This project is trying to bring the kyman keyboard mappings to Linux it is/will be GPL'd

Changes needed to your Linux distribution

The various Linux distros have different configuration applications for setting things like keyboard. These few notes give a quick guide as to what files you need to change and where you need to look.

Fedora/Red Hat

The application `system-config-keyboard` sets the keyboard for your setup. It is simply a configuration tool for the normal X keyboard mappings. The file **''keyboard_models.py''** which is part of **rhpl** needs to be edited to add your keyboard.

Creating keyboard layouts for Mac

SIL has created a nice tools called Ukelele designed to create the XML files needed by the Macs keyboard system.

1.14.6 Web-based Fonts Rendering Technologies

This is a list of resources that list web-baed font rendering technologies for various scripts. The idea is that if a person does not have a font that a technique of either font uploading or graphical glyph substitution is used to ensure that a user can still see the text.

Not all of them are useful, most of them probably don't apply to you as computers can probably render your script correctly. Some are non-Free but may give ideas to people.

Glyphgate

- http://www.glyphgate.com/
- http://www.nunatsiaq.com/news/nunavut/41008_11.html
- http://www.attavik.net

Mainly for web based rendering of languages without the need to install fonts. In the worst case scenario it will create images of all the missing glyphs so you can view the webpage without needing to install a font.

- *Unicode*
  - Unicode Checker on OSX provides system-wide Unicode codepoint and conversion/normalization services
  - Your Multilingual Mac is a comprehensive Unicode resource
- Fonts
  - The Unicode Font Guide
  - *Free and Open Source Fonts*
  - *Microsoft Core Fonts*
  - *Other fonts*
- *Keyboards*
  - How to edit X11 Keymaps or create your own
  - Edit or create OSX keyboard layouts using Ukelele
- Rendering

   – *Web-based Fonts Rendering Technologies*

1.15 Other localisation

1.15.1 Document translation

Translating documents can be quite different from translating software interfaces. Many issues specific to software localisation might not be relevant in documents, such as accelerators, translation length, constructed phrases, etc. However, document translation has several other issues that is good to be aware of.

Preparing for translation

Ideally a document should be prepared for translation. A good source document will make translation easier. Possibilities:

- Proofread the document (spelling, grammar, clarity)
- Use consistent terminology
- Read about "writing for translation" (1)
- For structured documents, use proper structure like headings and subheadings instead of using style only.

Translation

A lot can be said about translation in general, but this is only meant to give you some tips.

Be to be aware of issues arising out of translation memory. You could possibly have exact matches (identical string translated before), or In Context Exact (ICE) matches, where some translation tools will specifically indicate that the translation is identical, but also that the surrounding text from the paragraph is the same. It could also indicate agreement with regards to domain, file, date, etc.

Post-processing

After generating the translated document, you very likely need to do some post processing. Things to consider:

- Ensuring correct translation in cases where context might not have been obvious during translation
- Document layout, page layout
- Fonts or other styling changes
- Style of generated content, such as numbers
- Generated sections, such as Table of contents, list of figures, index, variables

1.15.2 Spelling Checkers

These are an important part of creating a complete language specific view of the operating system. Even English speakers prefer to see correct spell checkers for their locale (UK vs US vs South African). Certain languages by their nature do not need or cannot use the traditional wordlist type spellcheckers found on Linux.

Hunspell is the main spell checker on Linux[1].

There is a legacy of previous spell checkers on *nix platforms and these are:

- ispell – the original word based checker which includes affix compression.
- aspell – an enhancement of ispell with better algorithms for suggestions (newer versions have adopted the myspell affix compression)
- myspell – originally built for OpenOffice.org and previously also used by Mozilla it includes suffix compression (based on the ispell rules but in a new format) to produce smaller dictionary sizes. It runs on all platforms supported by OpenOffice.org and Mozilla.
- hunspell – an enhancement of myspell to allow more sophisticated language specific manipulation. It can use myspell dictionaries thus offering a smooth migration path. It is the default speller in OpenOffice.org, Firefox 3, Thunderbird 3 and Fedora 9.

If starting from scratch your best bet is to focus on a hunspell checker and make use of its language specific features if needed. If you maintain aspell and ispell checker you might want to migrate them. If you maintain a current myspell checker you should probably wait until hunspell infrastructure is widely deployed.

Resource

- Scandinavian spell checkers website with some useful tools.
- Debian Spellchecker packaging policies (Website)
- OpenOffice.org Lingucomponent
- Konjugator is a browser-based conjugator for Welsh verbs. This might be useful for developing your own spelling and grammar checkers in your language.

For spell checker development on OSX, these links might be useful. Just check that things are still current

- http://developer.apple.com/documentation/Carbon/Conceptual/UnderstandTextInput_TSM/tinptsm_intro/chapter_1_section_1.html
- http://developer.apple.com/samplecode/SpellingChecker-CocoaCarbon/listing3.html
- http://developer.apple.com/documentation/Cocoa/Conceptual/SpellCheck/Tasks/CreatingSpellServer.html

Web based corpus building

A corpus is a body of text used by language researchers and spell checker builder. You can find missing or new words for your spell checker by scanning the web. There are free tools that you can use to build your own web-based corpus. We developed CorpusCatcher for this purpose.

Other possibilities: corpusbuilder and text_cat (FIXME How to use these). The former searches the web using a public search engine and the later uses a statistical model to determine if the text found is indeed in your target language.

Once you have a list of potential words you can use the *new-words* script in src/wordlist in the Translate Toolkit SVN to identify words that are not in your language. Review these words and add them to you master wordlist.

Kevin Scannell's – An Crúbadán

- An Crúbadán home page

[1] Many distributions consolidated spell checking around Hunspell to some extent, for example Fedora, Firefox, Thunderbird and OpenOffice.org use Hunspell.

- Geez Crawler – a derivative of *An Crúbadán* that is crawling Tigrigna and Amharic

Language detection

This Python code could easily be used to develop language detection for a webcrawler: http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/326576

Other Crawlers

- http://home.hccnet.nl/r.j.baars/ (broken link) – used we think by the OpenTaal.org project

Letter Frequencies

The translate project has a simple python script that creates letter frequencies that can be used in the MySpell affix files TRY line. See translate/src/wordlist/letter-frequency.py in the Translate Toolkit CVS

Building

The easiest way to build your spellcheckers is to use our project spellchecker build framework. This will build MySpell and Aspell (Ispell temporarily disabled) spellcherckers from a common wordlist or wordlists. Look at the Afrikaans and Zulu dictionaries for a template of the process. Again this is in SVN in the *dict* module of the zaf project.

In more detail

Checkout the dict/ module from Subversion:

```
svn co https://zaf.svn.sourceforge.net/svnroot/zaf/trunk/dict dict
```

Directory layout:

- xx/ – the various language dictionaries
  - Makefile – various configurations for the spell checker building
  - myspell/ – myspell dictionary
    - xx_YY.aff – affix file
    - README_xx_YY.txt – installation and copyright notices
  - aspell/ – aspell dictionary
    - info.in – various configuration settings for aspell
    - Copyright – Copyright notice
- utils/ – the various utilities, generic Makefile.languages, aspell build routines, myspell build utilities.

Simple make instructions:

- `make` – makes all spell checkers
- `make myspell` or `make aspell` – makes the respective dictionary
- `make count` – gives spell checker word counts
- `make clean` – removes all autogenerated files

- `make wordlist` – create and packs the wordlists

Making it work

Make sure that your language is included in: http://cvs.gnome.org/viewcvs/gnome-spell/gnome-spell/dictionary.c

So that Gnome applications such as Evolution can make use of your aspell spellchecker.

Publishing

OpenOffice.org

To get the spellchecker onto the OpenOffice.org pages and thus downloadable from within OpenOffice.org. You will need to submit a bug report. Here is and example issue: http://www.openoffice.org/issues/show_bug.cgi?id=23201

ASpell

FIXME

Mozilla

Mozilla dictionaries must be tri-licensed (GPL/LGPL/MPL) for inclusion in the source tree, which results in inclusion in a language build. For many spell checkers this will probably be a problem.

Alternatively you can create a dictionary extension and upload it to Mozilla Addons. Users who upgrade Firefox are directed to the dictionary download page ensuring rapid adoption of your spell checker.

1.15.3 Evaluating spell checkers

spell_quality.py

The spell_quality.py script can be used to calculate basic spellchecker quality metrics.

It doesn't have an installer, and depends on a working installation of Python and the pyenchant library. Simply download spell_quality.py and run it with your python interpreter. It has only been tested with Python 2.5, but Python 2.6 should work fine. Please report on your success with Python 3.

usage:

```
./spell_quality.py --correct correctcorpus.txt \
   --incorrect incorrectcorpus.txt --language zu_ZA
```

The corpus files should include one single word per line and should be in the UTF-8 encoding. It can be prepared based on any given corpus with commands similar to

```
tr -c 'A-Za-z' '\n' | sort  > corpus.txt
```

or if you only want to test unique words:

```
tr -c 'A-Za-z' '\n' | sort | uniq > corpus.txt
```

False Nagtaive Frequency Tables

Frequency tables of words deemed incorrect by a spellchecker can be generated with a simple command line

```
hunspell -d zu_ZA -l corpus.txt | sort | uniq -dc | sort -nr | head
```

Metrics

To get an idea of the quality of a spellchecker, we can use the information retrieval metrics of precision and recall.

Precision: is a measure of the exactness of the spellchecker's responses, it basically tells you how much you should trust the spellchecker when it tells you this word is correct.

Recall: is a measure of the completeness of the spellchecker, it tells you how much of the language the spellchecker covers, the lower the value the more likely it is that the spellchecker will complain about correct words.

It might also be useful to measure the same useful to measure the same metrics but for the case where the spellchecker identifies the word as incorrect instead of correct.

In that case recall is called specificity and it tells you how likely it is that the spellchecker will catch all incorrect words.

Generally speaking the spellchecker should have very high Precision and specificity to be useful.

We also calculate accuracy which is derived from both precision and recall and is a general measure of the quality of the spellchecker.

The *spell_quality.py* script in the wordlist directory of the toolkit can be used to measure precision, recall, accuracy and specificity.

Actual performance

Now these metrics can be a bit too generic, to judge whether a recall of 30% is good or bad you need to know which 30% of the language does the spellchecker cover. Or more importantly which correct words is it going to assume are incorrect.

It's ok if the spellchecker doesn't know about obscure, archaic, deeply technical or other rarely used words, but it is a very big problem if it can recognize millions of words but fails to recognize a few widely used ones.

We can manually evaluate coverage by generating *False Nagtaive Frequency Tables*.

1.15.4 A spellchecker for your language

Does the current spellchecking software work well for your language? As software is translated into more and more languages, we are finding that writing aids software based on European-language structures is less and less effective.

So what can we do about it? It's time to review our current resources, and work together to create writing aids tools appropriate to a wider range of languages, or appropriate to specific languages. Some case-studies are discussed below.

This page will link to discussions on this subject in different mailing lists, to wiki pages detailing it for and in different languages, and to software being developed, or which can be used, to meet these needs.

Please add any information you think could be useful. :)

Case Studies

Vietnamese

Nguyn Thái Ngc Duy

I have been working on vspell, a Vietnamese spell checker, since 2003 (there were huge gaps when I did not work on it at all, though). The source code is available (click on the first "snapshot" link to get a tarball). The core idea is quite simple. It is trained with a word-segmented corpus. When a sentence is given (actually a phrase because it still does not understand "sentence"), it will generate similar sentences based on common spelling errors. It then uses statistics from the corpus to determine which sentence (the original one or one of the generated ones) is "better". If a generated one is better, then it assumes the original one is misspelled. That's all. The rest of work is matching the original one and the "right" one to see differences between them and tell users about that.

The result as of three months ago was not very promising: precise rate was about 60%-70% (I expected at least 80% to be useful). I was investigating to see why the precise rate was low, and had some technical difficulties.

I've been working on a spellng checker for Vietnamese. The problems are:

- It requires input as phrases, not words. As far as I can tell, hunspell and almost all spelling checkers take input as words.
- It uses a lot of memory (256MB for itself)
- It needs to be trained to be able to differentiate good phrases and bad phrases. There are some difficulties in my training method (I don't know, I may give up on my method, as it is becoming infeasible, something with number explosion).
- It's not stable. I spent most of my time training it and testing it with a small sample. It liked to crash back then ;)

The first problem means we must interact with Openoffice without Hunspell. That's not easy to do.

But the main problem is the third. My approach is statistics-based. It requires lots of (correct) word-aligned sentences to be trained on. That kind of corpus for Vietnamese does not exist (at least freely). So my workaround is to take a raw corpus and train repeatedly to get better result each iteration. The workaround has number explosion problem. It breaks 32-bit integer limit easily and also "long double" limit. In short, until I find a feasible training method, my spell checker is no use.

The basic problem in Vietnamese

Vietnamese "words" are often composed of more than one word. Words are usually monosyllabic, so we can think of them as syllables of these longer words. However, current spellchecking tools treat each Vietnamese "syllable" as a separate word. This means that when you make a mistake that is still a valid word, e.g. typing **màu hình** instead of **màn hình**, current spellchecking tools will still recognize « màu » and « hình » as valid separate words, and not detect the error.

Ivan Garcia

(Note: Ivan's current Hunspell spellchecking dictionary is being used in OpenOffice.org, and in Firefox and Thunderbird.)

Duy

I know my statistical method isn't ideal. A rule-based approach would be more realistic. But the rule-based one requires human power to build the ruleset. A rule generation approach like TBL (Transformation Based Learning) requires an annotated corpus, which I don't have.

Identifying composed words is also what I want my spellchecker to do.

I don't think a grammar-checker is viable, due to the complexity of Vietnamese grammar. My spellchecker is basically a spellchecker. Although it could also be able to detect some semantic/grammatic mistakes as well.

Even using current grammar-checking tools, you will have more troubles with Vietnamese ;) Before you discuss grammar, you must split a sentence into words (actually annotated words but that's not the point). It's already difficult to do that in Vietnamese. Now you are supposed to do that on a **misspelt** text. Good luck :D

European languages don't have this problem, as distinct words can be easily recognized. CJK languages do though, but I guess CJK spellchecker status in OOo is just the same as it is for Vietnamese.

To have an idea how hard it is to split a sentence into words, let's take a corner case: « **Ông già đi nhiu quá** ». You can understand this sentence in a couple of ways:

- An old man goes a long way (**Ông-già đi nhiu quá**, notice « **Ông già** » is a single word)
- He gets very old (**Ông già đi nhiu quá**, notice « **Ông già** » is two separate words)

Now suppose « **già** » is mistakenly written as « **dà** », then pass the sentence to a grammar checker ;)

Could we train a spellchecker?

Is it possible to train a spellchecker, as one trains a Bayesian spamfiltering program like SpamSieve on OSX? If we have a group of volunteers each building up a corpus, and training the spellchecker each time it makes a mistake, perhaps we could amass the data we need.

This discussion

You can follow this discussion on the specially-created anoi-spell mailing list, or on the Translate Toolkit mailing list.

Quechua

Amos Batto

It is interesting hearing about the difficulties of spell checking in Vietnamese. I have also run into problems using Hunspell to spellcheck Quechua (an indigenous language of the Andes), but of a very different nature.

Quechua is an agglutinative language. Most words have 1 or 2 suffixes, but some can have as many as 8 or 9 suffixes. Most suffixes have to be added in a very specific order, but a few can appear in almost any order. Needless to say, the possible combinations of suffixes is almost infinite and almost impossible to list in an affix file. When I tried to write out all the possible combinations, I got up to 500 pages of combinations and that was only combining 1, 2, and 3 suffixes.

Hunspell is much better than aspell and ispell because it allows an infix and 2 levels of suffixes, but it is still woefully inadequate for languages like Quechua.

Another problem of Hunspell is a lack of a "sounds like" feature, as is found in Aspell. In Quechua, K, K', KH, Q, Q', QH are all easily confusable letters, along with their Spanish equivalents: QU and C. In order to properly spellcheck in

Quechua a "sounds like" function needs to be added to Hunspell. The code for "sounds like" shouldn't be that difficult to write, but the Hunspell code looks pretty complicated and I haven't figured it out.

//Note: Kevin Scannell, who was one of the developers of Aspell contacted me to say that he intended to add the Aspell "SoundsLike" function to Hunspell. I hope that he finds the time to do this since it will help me greatly. Kevin also noted that the metaphone function in Hunspell could act like SoundsLike to some degree.//

If you have more interest in learning about the challenges of a Quechua spellchecker, see this note I wrote to the Hunspell developer explaining our difficulties.

Zulu

Friedel Wolff

The issue of agglutinative languages is quite interesting for me since we are working on spellchecking in Zulu which is also of this nature. We are starting to work now on a program to help people review word lists and to identify word roots. This is all done under the assumption that identifying roots is the most important part of the work in terms of the words and word lists, and combined with proper affix rules (developed separately), we can create a usable spell checker (for an agglutinative language).

People interested can have a look at our ideas for how this should work. This is currently really meant to be a small project that can be implemented quite quickly.

Statistical Support

Kevin Scannell can generate word frequency lists and other useful statistics for many languages (more than 400 as of May 2008) using his web crawling software An Crúbadán. There's a good chance your language is already supported if it has a non-trivial presence on the web. Contact him (kscanne at gmail dot com) if you are beginning development of a spell checker and plan on releasing it under and open source license.

Your language

1.15.5 Hyphenation

Resources

- A simple explanation of TeX based hyphenation – OpenOffice.org uses the same TeX files to hyphenate
- patgen – creates Hyphenation pattern files from a set of good hyphenations
- Frank Liang, Word hy-phen-a-tion by com-puter, STAN-CS-83-977, Stanford University Ph.D. thesis, 1983. – the creator of the Hyphenation system in TeX
- OpenOffice.org hyphenation page – OpenOffice.org use the ALTLinux hyphenator which is based on libhnj library by Raph Levien. It uses TeX hyphenation dictionaries with small corrections.
- Current OpenOffice.org hyphenation dictionaries
- ALTLinux standalone hyphenator
- Knuth's The TeXbook – you can't generate this but you can read the TeX in appendix H
- This patgen2 tutorial might be helpful
- This is a well commented (in TeX) version of patgen

- A very good explanation with clear examples for OPatGen (a Unicode enabled patgen)
- A very good set of tools written by the Dutch TeX team. Also useful for manipulation of word lists.
- Automatic non-standard hyphenation in OpenOffice.org written by László Németh about the improved hyphenation technology in OpenOffice 2.0.2 and later.

Creating you own hyphenation dictionary

Two approaches are possible to construct your hyphenation dictionary. An automated approach, and a manual approach. Which one is best for your needs, will depend on the characteristics of your language, and the existing information you have at your disposal.

If you have a list of correctly hyphenated words that is representative of your language, the automated approach might be best. If you don't have such a list, or it is not high quality, or the rules of your language is very simple, the manual approach might be best.

Take note of which programs and descriptions here refer to TeX format and OpenOffice.org format.

patgen

FIXME this is the process as I understand it now. It might be wrong!

You use patgen to create the initial hyphenation file.

```
patgen DICTIONARY PATTERNS OUTPUT TRANSLATE
```

Where:

- dictionary – is the input hyphenation list
- patterns – is the existing hyphenation pattern file
- output – is the new hyphenation pattern file
- translate – is a configuration style file (it is explained in some places but not that well)

When you run the command you will be asked various questions, again not sure of what these are about.

At the end you will have a pattern file that can be used in TeX.

By hand

This section explains the format of the hyphanation file required by OpenOffice.org. You might still be interested to understand how this works if you need to fix mistakes caused by another approach.

You need

- altlinuxHyph from http://lingucomponent.openoffice.org/hyphenator.html
- grep

Patterns are specified which describe patterns where hyphenation can occur and where hyphenation cannot occur. The priorities of breaking or non-breaking points are set by numbers (about 1-6). Odd numbers (1,3,5) indicate places where hyphenation can occur. Even numbers (2,4,6) indicate places where hyphenation are not allowed. All patterns that match a particular word are applied to a word. Higher numbers overrule the lower numbers. The places with odd numbers left indicate possible hyphenation points. A full stop can be used to indicate a word boundary (either beginning or end).

Consider the word "example". The word is prepared by putting a full stop in front and at the back. Therefore we now have ".example.". The hyphenation software searches the file for patterns that match any part of the prepared word. Lets say the following patterns match:

```
x1a
xam3
4m1p
1p2l2
```

This is how they are applied:

```
. e x a m p l e .
    x1a
    x a m3
       4m1p
         1p2l2
-----------------
. e x1a4m3p2l2e .
     ^ _ ^ _ _
```

Now we can hyphenate at the points of the odd numbers. Note how the even numbers overrule certain hyphenation points that otherwise might have caused wrong hyphenation. Therefore, "ex-am-ple". The software can now choose the hyphenation point that results in the best layout. There will usually be settings in the software that prohibits hyphenation in the very first and very last characters of a word, in order to avoid having just one or two characters left on the beginning or end of a line. (OpenOffice.org 2.0: *Tools* → *Options* → *Language settings* → *Language aids* → *Options*)

To encode exceptions, simply make sure that the pattern is surrounded by full stops. It might be necessary to encode exceptions in cases where a word doesn't follow normal spelling rules, for example with brand names, person names, etc. (.Kin1sa1s6ha.)

A confusing problem with the hyphenation patterns is the way they interact. Consider the following hyphenation file:

```
ISO8859-1
n1t
prin2t1able
```

Now consider the words "print", "printable", and "printer". Run "example" from altlinuxHyph as follows (each pair of lines contain both what was typed in, and the output):

```
./example printer.test /dev/stdin
print
print
printable
print-able
printer
printer
```

"Print" won't be hyphenated, because the hyphenation point specified by the first hyphenation pattern ("n1t") occurs to close to the end of the word. "printable" is hyphenated as we expect: "print-able". If we now input "printer", we are surprised: it doesn't hyphenate at all. This happens because the second pattern ("print2t1able") is matched first (from the "p" up to the "t"), but then discarded when the "e" in "printer" doesn't match the pattern. At that stage it doesn't go back to consider all patterns from the second character, but only continues from the last character that matched, "t". To solve this, edit the file as follows (add the last pattern):

```
ISO8859-1
n1t
```

(continues on next page)

(continued from previous page)

```
prin2t1able
prin1t
```

For the word "printer", the second and third patterns will both match up to the "t", but the second pattern will not be considered. The first rule therefore has to be "repeated" for the case where the second rule will mask it out.

It is important to remember that the beginning of word marker (".") will be handled like a normal character, and therefore the same masking problem can be obtained.

ALTLinux

ALTLinux make changes to the TeX hyphenation file that relate to optimisation and performance.

```
perl substrings.pl <tex hyphen file> <alt linux hyphen file>
```

Now add your languages encoding to the top of **alt linux hyphen file**. Now you are ready to include this in *OpenOffice.org*.

Including your hyphenation in OpenOffice.org

FIXME need to check this but this is just anecdotal based on my experience with MySpell

Your hypenations dictionary need to be included in dictionarl.lst the format is something like:

```
HYPH xh ZA hyph_xh
```

Where:

- HYPH – indicates its a hyphenation dictionary as apposed to a spelling DICTionary
- xh and ZA – the language and country
- hyph_xh – the name of the dictionary file without the .dic suffix

FIXME once again please check this :)

When the thesaurus function is activated in a program, it supplies the user with possible replacements for the highlighted word or word group. It can indicate the word types, synonyms, related words, abbreviations or complete forms, more desirable forms, etc.

1.15.6 OpenOffice.org

- Download MyThes
- create "dat" file with thesaurus information:

```
ISO8859-1
talented|1
(adj)|gifted|untalented (antonym)
talk|7
(noun)|talking|conversation (generic term)
(noun)|discussion (generic term)|discourse (generic term)
(noun)|lecture (generic term)|lecturing (generic term)
(noun)|lecture|public lecture|speech (generic term)
(verb)|speak|communicate (generic term)
```

(continues on next page)

(continued from previous page)

```
(verb)|spill the beans|let the cat out of the bag|keep quiet (antonym)
(verb)|lecture|teach (generic term)|instruct (generic term)
```

- run the indexing program that is part of MyThes
- add both the "dat" file and the "idx" file to the "dict" directory in the installation
- add a line like the following to "dictionary.lst":

```
THES en US th_en_US_v2
```

- restart OpenOffice.org (and the quickstarter in Windows, if used)

Select a word (like "talented" or "talk") and activate the thesaurus. The thesaurus can be accessed under *Tools → Language*.

1.15.7 Grammar Checkers

Kvein Scannel is developing a grammar checker called *An Gramadóir*.

Resources

- Project page
- Developer's manual

1.15.8 Applications that need non-translation localisation

These are applications that have country specific or language specific configurations. Ie there are things that you need to change to make the application more useful for user of your language or users in your country.;

The list merely highlights the issue but most of these have not been elaborated.

Country Specific

- KOrganizer – public holidays
- Evolution – public holidays
- Any Internet dialup logging
- Mozilla dates and times (might be locale specific)
- OpenOffice date and time formats (might be locale specific)
- KDE country and regions selector in locale
- KDE country.desktop file
- Mozilla region packs
- Default printer page sizes – should be set in locale
- Any weather application or user eg RSS feeds in Evolution
- Validate weather service namign conventions might be out of date
- Spellcheckers with:

    - Geographics names: province, cities, mountains, rivers, airports, harbours
    - People: statesman, hero's, common names
    - Animals: common names for fauna and flora
- City locations for things such as KStars
- Country specific shortcuts for Konqueror
- Mozilla shortcuts already built
- Mozilla local search engines and defaults already set

Language Specific

- KTuberling – language wav files
- Locale files
- Mozilla, Konqueror – default language preferences set

Both Language and Country specific

- OpenOffice – locale files

1.15.9 Localising Calendars

KOrganizer

KDE's KOrganizer stores country holidays in files designated holiday_XX where XX is the iso639 country code. In KDE SVN these are located in kdepim/korganizer/plugins/holidays/holidays/holiday_XX

1.15.10 Localising Weather Applications

There are a number of weather related applications for the Linux desktop. Most of these lookup up weather data from airport meteorological stations. These stations are identified by an international 4 character abbreviation. These applcications use this to lookup the correct weather information.

Applications

Localising GWeather

This email explains what needs to be done to localise gweather so that it has the right names for locations in your locale.

```
From: Davyd Madeley
To: gnome-love@gnome.org, gnome-i18n@gnome.org
Date: Tue, 30 Nov 2004 03:10:04 +0800
Subject: GNOME Lovers Needed: l10n work for locations database

gweather ships a very comprehensive locations database. GNOME-Applets
2.9.2 has some 3000+ more locations to this database[1] from syncing
with the latest METAR list, but now we need your help.
```

(continues on next page)

(continued from previous page)

```
The database now has support for  tags (as well as ,
 and ). While region, country, and state are all easy to
work out, city tags are hard, it's easy to figure out what locations are
in a city (sometimes) but not what their local names are. As a result,
many names are things like "Location 1" or "My Location (alternate)".

This is where you come in, spread around the world, and having a good
knowledge of local, state and maybe even national geography, you will be
able to clean up the locations file for your area.

How do you do this?
 1) Check out the latest version of Locations.xml.in [2]
 2) Fix up your area, the format for the XML file should be fairly
evident, or you can consult the DTD [3].
 3) Use xmllint and the DTD to check that the file structure is still
sane.
 4) File a bug in bugzilla under the gweather component of
gnome-applets. Give it the subject, "Locations Love - $my_area". Attach
a context diff of the Locations.xml.in file.

If you're a new contributor to GNOME, this is a really good way to get
yourself in the Changelog and NEWS file. This type of bread and butter
work is really easy, and you don't even have to build any software to do
it ;)

Remember that all strings in the Locations.xml.in file should be in the
C locale (US english). This means that locations should use their
American names or anglicized names. Otherwise use the local name. If you
use an anglicized name, add a comment to the XML so that translators
know about the local name.

I would really like this done as soon as possible so that translators
can get to work on getting the translations for the database up to
scratch. I want to have a really rocking locations database ready for
GNOME 2.10 (and eventually weather.gnome.org).

Also, maintaining the locations database has been proving to be an
increasingly time consuming task for me. If you feel you might be up to
scratch at maintaining this database, drop me a line, I could do with
some help.

Happy Hacking,
--davyd

[1] a log showing new locations/changed locations and removals is here:
http://oracle.bridgewayconsulting.com.au/~davyd/misc/locations_changes.log
[2] http://cvs.gnome.org/viewcvs/*checkout*/gnome-applets/gweather/Location=
s.xml.in
[3] http://cvs.gnome.org/viewcvs/*checkout*/gnome-applets/gweather/location=
s.dtd

--
Davyd Madeley              http://www.davyd.id.au/
```

Localising Evolution

Evolution Locations File

FIXME I think this no longer applies to Evolution 2.x

Create a country definition `/usr/share/evolution/1.4/Locations` and ensure it is included under your region. Add locations into your region file.

- *Localising GWeather*
- *Localising Evolution*

Problems

FIXME this data may be out of date please check and update it.

In Evolution there is a problem in that location names are not localisable. Bug #89746 refers to this problem.

Where do you get the data

More pressing then the localisability is the fact that many localitions are missing for countries. The data is easily retrieved. Use the following URLs to identify weather stations in yor country and locations.

You may have to correct a few where names have changed eg airport names or to reflect to a normal user not an aeroplane pilot where this site is located.

- https://www.notams.jcs.mil/common/icao/index.html – This is a good listing of sites sorted by country
- http://www.nws.noaa.gov/oso/site.shtml – Contains downloadable ASCII version and instructions on obtaining hardcopy versions – downloadable version seems to be missing
- http://www.wapf.com/world/FALA.html – Produces nice maps and proximity to other stations to help you identify the location.
- http://www.partow.net/miscellaneous/airportdatabase/ – A database representation of the ICAO codes.
- http://weather.noaa.gov/weather/metar.shtml – You can retrieve data from multiple stations. Useful to check whether the ICAO code is valid for weather data.
- http://weather.noaa.gov/cgi-bin/mgetmetar.pl?cccc=FAJS – The actual URL used by *Localising GWeather* to retrieve weather data.

Please also check the data of the weather, it might be so out of date that it is not valid.

1.15.11 Localisation of KStars

KStars is a Desktop Planetarium for *KDE*. It provides an accurate graphical simulation of the night sky, from any location on Earth, at any date and time. The display includes 130,000 stars, 13,000 deep-sky objects,all 8 planets, the Sun and Moon, and thousands of comets and asteroids.

You can localise KStars by adding cities for your country to the file Cities.dat.

- *Document translation*
- Language Tools
  - *Spelling Checkers*

        * *Evaluating spell checkers* – how to evaluate the performance of spell checkers
        * *A spellchecker for your language* – Documenting a discussion on developing spellcheckers that meet the needs of different languages
        * Enchant
        * Hunspell
        * Aspell
        * CocoAspell integrates the Aspell dictionaries into the OSX system-wide spellchecker
    - *Hyphenation*
    - *OpenOffice.org*
    - *Automatic correction*
    - *Grammar Checkers* checkers
- Other
    - A list of programs with *Applications that need non-translation localisation*, non-translation, localisation requirements
    - *Localising Calendars*
    - *Localising Weather Applications*
    - *Localisation of KStars*

1.16 Notes to programmers

1.16.1 Working with Gettext

This is not a replacement of the Gettext manual.

As translators we have written these pages to highlight feature of Gettext that, if used, make our lives so much easier. So please read this realising that if you follow these ideas you help us improve the localisation of your application. And we'll bug you less!

Report-Msgid-Bugs-To

Reason

As a translator you often have no idea where, how or to whom to report an error found in the source string. Sometimes we need to tell you to correct spelling or grammar. We need to ask what the context is or we need your help in defining a term we do not understand. Without a correct email or URL here we have to guess how to communcicate and spend a large amount of time on the wrong lists trying to help make the software better. Please fill it in.

Usage

The gettext documentation says:

```
Report-Msgid-Bugs-To
  This has already been filled in by `xgettext'.
```

The developer thus needs to pass the option:

```
--msgid-bugs-address=...
```

to `xgettext`. With recent gettext infrastructure this is a field in `po/Makevars` that the developer fills in; the rest is automatic.

1.16.2 Plurals

What is a plural? 1 file, 2 files, 3 files: file is singular, files plural. In this simple example we see the English plural form. Other languages have simpler or even more elaborate plural forms. Both KDE and Gettext have a method of managing plurals for PO files so that plurals can be used in languages that do not follow the English convention.

Why do you need this system?

Why not just write a piece of code like this:

```
if ( n == 1 ) then
    print "1 file"
else
    print "%d files", n
```

Well the simple answer is that not all languages follow the pluralisation format of English. Some use the singular form for both 0 and 1. Others have no singular/plural form. And yet others have complex systems that have at least 4 plural forms. For English speakers think of it as the different suffixes you see here: "1st, 2nd, 3rd, 4th … 21st, 22nd, 23rd, 24th…"

So in order to cater for the various plural forms a system was developed.

When to use plurals in your application

If you have numbers in the singular and plural form then use the plural construct:

```
"%n file"
"%n files"
```

However, if you do not have number in the construct then rather keep them as separate strings:

```
"Save the file"
"Save the files"
```

What do plurals look like in the PO file?

On KDE with a 3 plural form language:

```
msgid ""
"%n file\n"
"%n files"
msgstr ""
"%n form 1\n"
"%n form 2\n"
"%n form 3"
```

Gettext with a two form language:

```
msgid "%n file"
msgid_plural "%n files"
msgstr[0] "%n form 1"
msgstr[1] "%n form 2"
```

Also in Gettext style plurals the PO file header has an entry for the plural form which defines the number of plurals as well as the mathematical function to determine which one to use depending on the input number. Here is an example which would work for English:

```
"Plural-Forms: nplurals=2; plural=n == 1 ? 0 : 1;\n"
```

How to use the plural construct

KDE

FIXME this looks set to change for KDE4 with the Gettext method most probably being adopted

```
i18n( "One item", "%n items", count);
```

Note that because KDE uses a n to split the different plural forms you cannot use n in the actual message. To work around this you can use KDE's RichText features and the tag.

Gettext

FIXME unsure how this is used

1.16.3 Variables and the localiser

To a programmer they're easy, to a localiser well you'll see. . . For a great article on these issues see Text Fragmentation and Reuse in User Interfaces by Richard Ishida

Commenting

A message that lists 3 variables but doesn't tell anything to the translator about what the strings will contain does not help a translator. Even if you think the contents of the variable are self explanatory think again.

Here is an example of a poorly commented message:

```
msgid "%s returned %s after %s"
```

Your guess is as good as mine as to what each of those contain. Yes the previous and next string might help to determine what it means. But we want it translated so lets help the localiser.

A better implementation would have the following:

```
# %s - a URL
# %s - the error message
# %s - time as HH:MM:SS since the command was executed
msgid "%s returned %s after %s"
```

This explains exactly what each variable will contain.

Allowing reordering

English speakers assume that language order will not change. In many languages the order of sentences will be different from that of English. Many systems allow for variables to be reordered. Please ensure that your variables can be reordered.

A fictitious translation of the above examples may appear like this:

```
msgid "%s returned %s after %s"
msgstr "Returning %2s was %1s after %3s"
```

As you can see by allowing reordering the translator can convert the message in the structure of their language.

Check with the implementation of Gettext (or other localisation framework) on how to allow for reordering.

0 or O (zero or oh)

Don't confuse translators, choose numbers or letters that cannot be confused with other letter or numbers. In OpenOffice.org there is a section that has a variables %O (% oh) which almost all translators translate as %0 (% zero) as that is the more logical option for them, they're used to %1, %2, etc. If they do not have a font that makes it clear that this is a letter then they won't see it and they break the translation.

1.16.4 Translation Comments

Why do you need translation comments?

Consider this simple case; you have a menu item called "Manual". You know what it means, but when the translator sees this they will wonder did you mean:

- a document or help manual, or
- a manual process?

This is the simplest case where a translation comment such as "The installation manual" helps to clarify the situation and makes a translator more productive.

More examples of the need for translation comments

Real world examples are best. This is a discussion over the use of the word "Forward" in Northern Sotho:

"When you go forward. You go 'Pele', But when you forward the document, O 'Fetišetša pele', so if you just say forward, we don't know what you are talking about, it is better if it is in a sentence. But in this case i think we will use 'pele' because on the string no:86 and 88 there is "show previous page in history" and "show next page in history"

Was there translators guess correct? I think so, but it makes it so much easier if they don't need to be super sleuths as well as translators.

What should be commented?

Ambiguous words

Words that can be used as a noun or verb in English but may have different names in another language

Environment variables, Config files entries

Is it really obvious that TERM should not be translated?

Filenames

Filenames that you do not want translated eg .Xauthority. Especially if they appear translatable.

Program Names

Some sentences do not make it clear that the text refers to a program name which you probably don't want to translate.

Protocols

Especially when they appear as URIs eg help:index. Also protocol responses eg. "Server returned HELO" very tempting to translate. Or "KEEPALIVE response". Should KEEPALIVE be translated?

Types of comments

The types of comments depend very much on the localisation system implemented:

KDE style comments

The appear to the translator like this:

```
msgid ""
"_: The installation manual\n"
"Manual"
```

One advantage is that they can double as context strings. In Gettext PO files you can only have one translation of the source English word. So if your language translates an English word differently depending if it is a noun or a verb then you will have a problem. Context information in a KDE style comment essentially means that the strings are different and thus allows you to distinguish between verb and nouns spelled the same way in English.

Gettext comments

These appear in the # comments of a message block. They are sometimes overlooked by translators as usually the PO editing tool does not make them obvious.

Adding comments

FIXME not sure how to do it please refer to the Gettext manual and KDE programmers manual

1.16.5 Unicode normalization

This mostly affects programmers wanting to have search functionality in their application.

What is normalization?

A composed character in Unicode can often have a number of different ways of representing the character. E.g

- Precomposed –> U1e3c
- Composed = L + ^ –> U004c + U032d

You'll notice that if you where comparing the above two characters that indeed !=

In order to do correct comparison the characters need to be normalized, thus they need to be reduced to the same character composition.

Normalization in my programming language

The following show how to normalize your data in various programming languages.

In most cases you don't want to alter the actual stored data but just want to normalize when comparing data, then throw it away.

Python

```
import unicodedata

string = unicodedata("NFC", string)
```

You can replace NFC with NFD, NFCK or NFDK. Read the Python docs for the unicodedata module for more detail on what those options mean. For most cases it is enough to use the above to bring the data into the same form and perform your comparison.

1.16.6 String QA

The quality of translations is proportional to the quality of your original strings. You can ensure higher accuracy of translations by adding contextual information, but the original string itself is the source of the translation. If you put the same care into constructing your original strings as you do into your coding, you will have effective and intuitive translated software.

Checking

Just as you always read over a line of code before running it, always read over a string before using it. That single check will save you a lot of errors. Spellchecking is also useful in catching typos. You can spellcheck PO files using pofilter.

Reports

You can also receive reports from translators who have spotted typo or other errors in your original strings. Translators are a useful resource for this kind of error.

After fixing

Once you make typo or grammar fixes in your original strings, unless you have actually changed the meaning of those strings, you want to **unfuzzy** them. Strings are automatically marked fuzzy on update when the original text has changed. This is useful when updating translations, but in the case of typo/grammar fixes, it means that **every** translator for **every** language will have to check **every** string you've fixed. This is a huge waste of resources, and thus a very unpopular situation with translators. Imagine having to check every line of code because someone had fixed the spelling in a comment.

Howto Unfuzzy

So, good i18n practice is to **unfuzzy** strings after changes that don't alter their meaning (e.g. typo/grammar fixes). How do we do that?

Oddly enough, since every project does this regularly, there doesn't seem to be a quick and effective method. All we're really doing is diffing the changed files with the previous files, then removing all fuzzy tags which have only appeared in the changed files. So it could be done with pogrep: if you work it out, please add the command here.

Meanwhile, the Debian project has its own method, and there are a couple of useful items in the gettext manual (Fuzzy Entries PO Mode Index).

Gettext and Po4a

Nicolas François adds:

In some specific cases, i.e. when all the PO files fully translated before the typos are fixed, msgattrib (gettext) is very fast and very safe.

The po4a package has a tool, msguntypot, which tries to deal with the generic case, but the process to use it (8 steps) is quite complicated and thus error-prone. msguntypot is not yet well-tested either, and all files should be backed-up before using it.

Unless you need to use msguntypot, I would recommend unfuzzying only complete translations and using msgattrib.

Translate Toolkit

Friedel Wolff adds:

If you want to unfuzzy all fuzzies in a single file, this should work:

```
msgattrib --clear-fuzzy input.po -o output.po
```

If you want to work recursively on many files, or want to select the messages on which to do the unfuzzying, using pofilter/pogrep with pomerge (all Translate Toolkit programs) might be more powerful.

1.16.7 Monolingual file formats

Monolingual files are files that contain only one language (either the source language, or the translation). Most documents, including plain text files are such documents. They are usually written for another purpose that only needs the one language, and is then replaced entirely with a translated version.

Such files are sometimes used for localisation. It is found in:

- *The Mozilla project* (Mozilla DTD format, Mozilla and Java properties files, and others)
- Websites (HTML) and often in Wiki Syntax pages

- Java projects (mostly Mozilla and Java properties files files)
- Applications for OSX and iOS using strings files
- Adobe Flex applications
- .ini files, such as those used for the translation of Innosetup installation files
- Video subtitles
- Windows .rc files
- Document translation (such as for OpenDocument Format or DocBook)
- Symbian translation files
- PHP files used for translations

Monolingual files are contrasted with multilingual files that contain the original source text and the translation in one file. Examples include PO Files, XLIFF, and Qt .ts.

Issues

Monolingual files for translation present some issues that need to be solved in any non-trivial translation task.

Tool support

Many translation tools provide support for translation file formats such as XLIFF, PO or others, but only some monolingual file formats that the product chose to support. While many monolingual files can be edited in the tool that originally created the file (such as a word processor), such a tool might not be well suited for translation. Particularly, it might have limited or no support for terminology, translation memory, quality checks, etc.

Furthermore, for any given translation format, there are probably a selection of translation tools available, from which a translator or team can select the most appropriate for the circumstances. For many monolingual formats, only the original authoring tool is available, and provides no choice to translators who might need specific tools for their language or team.

Format issues

In the case where no tool or editor is available, translators are exposed to the raw file format, which might be XML or some other format with extra information. Such formats are often sensitive to mistakes which can render the whole file useless and causes an unnecessary debug process just to get the file in a usable order. Issues related to XML tags, escaping and correct formats are not always known to translators, and any time spent on these issues just mean less time for translation and review. Translators might now also have to worry about line-endings, BOM markers – things that are not even visible in normal text editors.

Another consequence of such formats, is that they are usually extremely badly suited for right-to-left languages like Arabic and Hebrew and provide an extremely frustrating translation experience where the markup or identifiers causes complexities with the text now being a mix of left-to-right and right-to-left text.

Another issue for languages in other scripts than the Latin alphabet, is that special fonts are required for their own language, but they might not necessarily have good support for the source text, which can affect readability of the source text.

Estimation

If you don't have your translation work in a translation file format, it might be hard to quickly get an idea of the size of work required to complete the task. Tools might be able to count strings, words and more for the translation formats that they support.

File updates

While you might be able to translate something the first time from the source text, if the source text is changed, it becomes significantly harder to update your translation to reflect all additions, removals and changes in the source text. This is where a translation format or translation memory is required to perform your update in reasonable time, and in fact, makes this a very easy operation to perform repeatedly (even automatically).

Metadata

Monolingual file formats translators encounter are seldom intended for translation, and therefore lacks support for a lot of meta-data that translators find useful. Examples:

- Comments from a programmer or original author
- Comments from translators
- References to where the text is coming from
- Information about previous or alternate translations
- Translation state, such as "fuzzy", "reviewed" or similar

Quality Control

While review and quality assurance is often possible for several translation formats, it is hard to do any formal type of quality control for files that don't have tool support. It might also be hard to compare your translation to existing practices in any automated kind of way.

Reuse

When using a custom tool, it is often difficult or impossible to reuse translations from previous translations, and also to reuse your current translations in other projects.

Solutions

Working with monolingual file formats is very common and standard solutions exist. The monolingual formats are obviously used in many cases as the appropriate or only formats for documents, web pages, or certain localisation systems.

Convert to translation formats

This approach assumes that all translation activities will happen in a translation format such as PO or XLIFF. In this way, a tool converts the monolingual file to a translation format, and all activities happen in the translation format.

Afterwords the translated file is converted to a monolingual file in the target language, possibly with help of the original source file.

This has the advantage that it is easy to generate the target file even if the translation isn't finished, without having to soil your translation with the source text that you need to fish out later. For software localisation, this means you can easily translate your program bit by bit, but keep testing it even if you have only reached 20% so far.

This is the approach taken by the Translate Toolkit, with all the converters that convert to PO or XLIFF formats. This way the powerful translation tools such as Virtaal, Pootle and all the tools in the Translate Toolkit can work on rich translation formats that have all the functionality that translators expect.

An added advantage is that the translation format can be saved in a version control system for reference, and all meta-data can be saved along with the translations.

Translation Memory

Although translation memory is a standard tool with translation in any workflow, an approach taken by some tools to solve the problem of monolingual file formats make extensive use of translation memory for the solution.

The solution involves "retranslating" the new monolingual file each time by means of the translation memory that contains the previous translations. In this case your translation memory needs perfect support for identifying the correct segment to reuse, and possibly use techniques such as in-context-exact matching.

This technique is not followed by the Translate Toolkit, although some other tools follow this method. It is not clear to what extent this method allows for powerful meta-data to be associated with translations for the benefit of translators in future.

1.16.8 Translating non-PO files using Pootle

We all have our favourite format, and you can certainly translate it on Pootle! Pootle is your own personal translation environment: we want you to be able to set it up to suit your own workflow and needs.

We also want to simplify the file-format issue for you. Whether you're translating application files, documentation, manpages or working with glossaries and translation memory, you can convert the file to your favourite format.

This will be a feature of the Pootle interface: simply choose your format and convert the file! The file converters are those in the Translate Toolkit, plus those in the po4a package, and any others our users find useful.

So, until you can use the Pootle interface to convert your files, simply email the Pootle mailing list, attach your file and ask for it to be converted to the chosen format and loaded onto Pootle! Then you translate it, and request conversion back to the original format.

I tested this out with my first manpage translation, where the Pootle admins converted my (n)roff file to PO format, I translated it on Pootle, then they converted it back to (n)roff, and the process worked very well. It was quick and accurate. There were no conversion problems or artefacts, and the resulting file works well.

So, what formats can we handle? Below is our list, but note that not all formats can be converted in both directions yet. If you want to add other formats to it, please edit this page or write to our mailing list. If you know of another filter which would be useful, please tell us.

After all, Pootle is what we make of it. :)

File format	filetype	Translate Toolkit	po4a
Comma-separated values	CSV		
Dia diagrams (uncompressed)			
Docbook	xml		
Document Type Definition	dtd		
Guide	xml		
Hypertext Markup Language	html		
Kernel configuration help files			
LaTeX files			
Manpages			
Mozilla translation project			
Nannoblogger files	nb		
Open Office translation project			
POD data	pod		
Portable Object Template	pot		
Standard Generalized Markup Language	sgml		
TeX documents and derivates			
Translation Memory eXchange	tmx		
Qt Linguist	ts		
OpenOffice.org Writer	sxw		
Text file	txt		
XML Localization Interchange File Format	xliff		
eXtensible Markup Language	xml		

Information on how to handle specific non-po files can be entered below.

OpenOffice.org

If you're starting a translation of OpenOffice in your language, you can take the .po files from http://sourceforge.net/projects/translate. If you have already started translation of OpenOffice with strings extracted from the source, you have to do the following:

- convert the OO format to PO format (see above)
- … more to come

(see oo2po if you want to convert them using the translate toolkit)

FireFox

- Other guides
  - KDE Programmer's i18n howto
- Working with Gettext
- *Plurals*
- *Variables*
- *Translation Comments*
- *Unicode normalization*
- intltool – used by Gnome and others to localise .desktop, .xml, .glade and other file types.

- *String QA* typo/grammar fixes to original strings
- *Monolingual file formats* formats create certain problems that need resolving for localization
- *Translating non-PO files using Pootle* – working with non-PO formats

Footnote

CHAPTER 2

Glossary

Note: many of the terms use an abbreviation of the form x10y where 10 stands for the number of letters between the first letter x and the last letter y (e.g. l10n)

a12n see Africanisation

a11y see Accessibility

Accessibility This is the effort to make software device independent, for input, output, and "display". this includes screen reading software, devices controlled by brain waves, and everything in between. This effort is mostly aimed at ensuring that computers and devices are usable by people with disabilities.

Africanisation Any activity that assists with the localisation and globalisation of software for the African continent.

CAT Computer Aided Translation. Any one of a number of tools that assist translators during translation.

CLDR see Common Locale Data Repository

Common Locale Data Repository this is an effort to provide a unifies locale data repository, thus merging a number of separate efforts. They have also defined an XML standard for storing locale data seems to be much more flexible then current methods.

fuzzy translation This has two meanings:

1. In Gettext translation: any string that has been modified or guessed by the Gettext tools ie a piece of text that needs your review.
2. In the localisation industry: a fuzzy string is an attempt to match the untranslated string with information from a TM. Usually this is expressed as a percentage eg 80% match.

g11n see Globalisation

Gettext A set of tools used primarily on Linux and Unix computers that makes a program translatable. The toolset provides tools to extract messages from programs, to manipulate the translations, to compile the translations and to allow a program to appear translated when it is run.

Globalisation FIXME

i18n see Internationalisation

Internationalisation this is the process of making a program localisable. This would cover work to allow for different fonts and character sets to be displayed correctly. To allow for scripts that run from right-to-left, etc. This work is usually done once for a language, once the toolkits and programs support a language the internationalsiation work is usually complete.

l10n see Localisation

l12y see Localisability

l18n see either i18n or l10n. This is a common misspelling of one of these words.

Locale this is a file that defines the locale data for your language and country. A locale covers things such as: the names of months and days of the week, how to write dates in your language, how your write number and currency values, the sort order of your language and much else. This data is essential for a computer to support your language.

Localisation This is the process of taking an existing program or operating system and making it work in your language. This would include translating: interface and documentation, creating a locale file and the creation of fonts. It would not include changing the widget set or program operation to cater for your language, that work would fall under internationalisation.

Localisability This is the degree to which a program can be localized – for example, fixed strings in a program that cannot be translated would be a localisability issue

m17n Multilingualization

Machine Translation when a machine translates from one language to another. You might also hear of Machine Assisted Translation, this is where a machine does the initial translation usually drawing heavily on a Translation Memory and a human translator does the final approval and correction.

MO Machine Object file. This is the result of compiling a PO file using Gettext's msgfmt command. MO file are used because they are much quicker for a computer to read then PO files.

MT see Machine Translation

Multilingualization This has one several different usages:

- The UI has a user selectable choice of languages.
- The user can input data in a number of different languages.
- The user can select the output in a number of different languages.

PO Portable Object file. These are the files produced by xgettext (cf.) that a translator translates into their language. These files are compiled using Gettext (cf.) tools into an MO file (cf.)

PO Editor Any one of a number of tools that can edit PO files. PO files are plain text files and in most cases a PO editor will hide the complexity from the translator and provide extra features like translation memory, etc.

TEnT Translation Environment Tool. A term suggested by some (Jost Zetzsche) as an alternative to CAT (cf.).

TM see Translation Memory

TMX see Translation Memory Exchange format

Translation Memory Exchange format TMX allows you to export, import and thus exchange translation memory data between various CAT tools.

Translation Memory to reuse existing translations you make use of a TM. It makes translation faster but also ensures the translations remain consistent.

Unicode This is a standard that is able to represent all characters for all alphabets in one character set. By doing this you are able to display any all and any character or language on one page. Unicode only provides the encoding you would still need fonts to display the text.

UI User Interface

UTF-8 A method of encoding Unicode using 8 bits. Other methods include UTF-7, UTF-16 and UTF-32. UTF-8 is the most dominant method of encoding Unicode characters, but UTF-16 is becoming more common.

XLIFF A new file format for representing translation data. It provides much more useful features then any current format and you will see this become more widely adopted in future.

xgettext a program which is part of the Gettext (cf.) tools that extracts translatable content from programs and stores them in a PO (cf.) files that will be translated by a translator.

CHAPTER 3

Language Information

3.1 Display settings

On this page, we're putting together a list of display settings that are needed or preferred for each language. If your language isn't represented, or if the information is incomplete in some way, please edit it or add to it. Some tips for developers are available at the developers/styling guidelines.

For many languages written in Latin scripts without diacritics, the default settings are probably OK.

List the fonts in order of preference, even if they are not widely available. Browsers will take the first font in the list that is available. If a certain glyph (character) is not available, it will continue in the list trying to find a font to display the glyph. Please specify your choice of Serif or Sans with the font name where relevant.

If a font-size adjustment is necessary to display your language readably, input it as a percentage relative to 100% (e.g. 130% for a size of 30% larger than the default). Input a percentage which should create a font size equally readable with the ordinary Latin script at the default size. Remember to take your resolution, browser settings, etc. into account. Your browser's setting for the minimum font size should override a smaller font size, so please make sure your browser preferences don't overrule what you specify here.

Underlining hyperlinks is a common practice, but a nuisance for some languages, especially those which have a diacritic or mark of some kind underneath the character. Indicate the ideal link-decoration for your language. If underlines don't work, perhaps overlines do, or links becoming bold when the mouse hovers over them. Colour-specific decoration is probably not ideal here. In any case, the mouse in your browser will usually change to a hand when it passes over a link, so links don't **have** to be marked otherwise.

If your language uses a different number system (such as many languages from the middle and far east), you can indicate it. Please indicate if not all readers of the language prefer the number system (like for Arabic).

If you need to specify another setting, add a column to this table and input the information. To specify the special characters for your language that Pootle should provide during editing, see Special characters.

ISO	English name	Fonts in order of preference	Font size adjustment	Hyperlink decoration	RTL	Number system
ar	Arabic	Tahoma, Nazli, DejaVu Sans			Yes	Some prefer Arabic-Indic digits
bn	Bengali	FreeSans, Muktinarrow, Vrinda	110%, line-heigt:120%	?	No	…
fa	Persian	Terafik, Traffic, Roya, Nazli, Nazanin, sans	120%		Yes	Eastern-Indic digits
el	Greek	Dejavu			No	
he	Hebrew				Yes	
ks	Kashmir				Yes	
ps	Pashtu				Yes	
ru	Russian	DejaVu Sans, Liberation Sans, FreeSans, Sans			No	support dashes and hyphens properly
ug	Uyghur	UKIJ Tuz Tom, Alkatip, Uyghur Ekran, Tahoma, Segoe UI, Microsoft Uighur			Yes	
ur	Urdu	Nafees Nastalique, Nafees Web Naskh			Yes	
vi	Vietnamese	Lucida Grande, Vu Phu Tho, URWVN fonts, DejaVu Sans	None	No underline, no overline, change colour on hover	No	Arabic digits, comma is decimal separator, dot is thousands separator
yi	Yiddish				Yes	
zh	Chinese		120%	Underline	No	
fr	French	Arial			Yes	

3.1.1 External links

- http://wiki.freedesktop.org/wiki/Software_2fFonts

3.2 Language Names

How do *you* write the name of your own language?

We're putting together this list as another helpful resource for translators. Please add your language to the table. The table is in alphabetical order based on the English name.

English	Original
Afrikaans	Afrikaans
Arabic	
Basque	Euskara
Brazilian Portuguese	Português do Brasil

Continued on next page

Table 1 – continued from previous page

English	Original
Breton	Brezhoneg
Bulgarian	
Catalan	Català
Chinese (simplified)	
Chinese (traditional)	
Cornish	Kernewek
Croatian	Hrvatski
Czech	Čeština
Danish	Dansk
Dutch	Nederlands
English	English
Finnish	suomi
French	Français
Friulian	Furlan
Galician	Galego
German	Deutsch
Greenlandic	Kalaallisut
Hebrew	
Icelandic	Íslenska
Indonesian	Bahasa Indonesia
Italian	Italiano
Japanese	
Javanese	Basa Jawa
Kazakh	
Korean	
Lao	
Malay	Bahasa Melayu
Malayalam	
Maori	Māori
Ndebele, South	IsiNdebele
Northern Sotho	Sesotho sa Leboa *or* Sepedi
Papiamento	Papiamentu
Persian	
Polish	Polski
Portuguese	Português
Romanian	Română
Russian	
Serbian	*or* srpski
Sinhala	
Slovak	Slovenský
Sotho, Southern	Sesotho
Spanish	Español
Sundanese	Basa Sunda
Swati *or* Swazi	Siswati
Tamil	
Tsonga	Xitsonga
Tswana	Setswana
Ukrainian	
Uyghur	

Continued on next page

Table 1 – continued from previous page

English	Original
Venda	Tshivena
Vietnamese	Ting Vit
Xhosa	IsiXhosa
Yiddish	
Zulu	IsiZulu
Nepali	Nepali

3.3 Plural Forms

This is a list of the plural forms, as used by Gettext PO, that are appropriate to each language.

If your language isn't represented – please add it, or if the information is inaccurate or inadequate in some way – please edit it. The Plural information is usually very hard to find and also in many ways hard for a new localiser to understand. So please see this as a repository that can help localisers. Understanding how the Gettext functions use plural forms will help you design a correct plural form.

Note: Launchpad also has plural information for many languages, please add it here if your language is missing.

Note: Mozilla now also uses plural forms. Although they follow a slightly different form, the underlying equations seem to be the same as those used by Gettext.

Note: Unicode CLDR also collects plural information for many languages (also in XML form).

ISO	English name	Plurals header in .po files
A		
ach	Acholi	nplurals=2; plural=(n > 1);
af	Afrikaans	nplurals=2; plural=(n != 1);
ak	Akan	nplurals=2; plural=(n > 1);
am	Amharic	nplurals=2; plural=(n > 1);
an	Aragonese	nplurals=2; plural=(n != 1);
anp	Angika	nplurals=2; plural=(n != 1);
ar	Arabic[1]	nplurals=6; plural=(n==0 ? 0 : n==1 ? 1 : n==2 ? 2 : n%100>=3 && n%100<=10 ? 3 : n%100>=11 ? 4 : 5);
arn	Mapudungun	nplurals=2; plural=(n > 1);
as	Assamese	nplurals=2; plural=(n != 1);
ast	Asturian	nplurals=2; plural=(n != 1);
ay	Aymará	nplurals=1; plural=0;
az	Azerbaijani	nplurals=2; plural=(n != 1);
B		
be	Belarusian	nplurals=3; plural=(n%10==1 && n%100!=11 ? 0 : n%10>=2 && n%10<=4 && (n%100<10 || n%100>=20) ? 1 : 2);
bg	Bulgarian	nplurals=2; plural=(n != 1);
bn	Bengali	nplurals=2; plural=(n != 1);
bo	Tibetan	nplurals=1; plural=0;

Continued on next page

Table 2 – continued from previous page

ISO	English name	Plurals header in .po files
br	Breton	nplurals=2; plural=(n > 1);
brx	Bodo	nplurals=2; plural=(n != 1);
bs	Bosnian	nplurals=3; plural=(n%10==1 && n%100!=11 ? 0 : n%10>=2 && n%10<=4 && (n%100<10 || n%100>=20) ? 1 : 2);
C		
ca	Catalan	nplurals=2; plural=(n != 1);
cgg	Chiga	nplurals=1; plural=0;
cs	Czech	nplurals=3; plural=(n==1) ? 0 : (n>=2 && n<=4) ? 1 : 2;
csb	Kashubian	nplurals=3; plural=(n==1) ? 0 : n%10>=2 && n%10<=4 && (n%100<10 || n%100>=20) ? 1 : 2;
cy	Welsh	nplurals=4; plural=(n==1) ? 0 : (n==2) ? 1 : (n != 8 && n != 11) ? 2 : 3;
D		
da	Danish	nplurals=2; plural=(n != 1);
de	German	nplurals=2; plural=(n != 1);
doi	Dogri	nplurals=2; plural=(n != 1);
dz	Dzongkha	nplurals=1; plural=0;
E		
el	Greek	nplurals=2; plural=(n != 1);
en	English	nplurals=2; plural=(n != 1);
eo	Esperanto	nplurals=2; plural=(n != 1);
es	Spanish	nplurals=2; plural=(n != 1);
es_AR	Argentinean Spanish	nplurals=2; plural=(n != 1);
et	Estonian	nplurals=2; plural=(n != 1);
eu	Basque	nplurals=2; plural=(n != 1);
F		
fa	Persian	nplurals=2; plural=(n > 1);
ff	Fulah	nplurals=2; plural=(n != 1);
fi	Finnish	nplurals=2; plural=(n != 1);
fil	Filipino	nplurals=2; plural=(n > 1);
fo	Faroese	nplurals=2; plural=(n != 1);
fr	French	nplurals=2; plural=(n > 1);
fur	Friulian	nplurals=2; plural=(n != 1);
fy	Frisian	nplurals=2; plural=(n != 1);
G		
ga	Irish	nplurals=5; plural=n==1 ? 0 : n==2 ? 1 : (n>2 && n<7) ? 2 :(n>6 && n<11) ? 3 : 4;
gd	Scottish Gaelic	nplurals=4; plural=(n==1 || n==11) ? 0 : (n==2 || n==12) ? 1 : (n > 2 && n < 20) ? 2 : 3;
gl	Galician	nplurals=2; plural=(n != 1);
gu	Gujarati	nplurals=2; plural=(n != 1);
gun	Gun	nplurals=2; plural=(n > 1);
H		
ha	Hausa	nplurals=2; plural=(n != 1);
he	Hebrew	nplurals=2; plural=(n != 1);
hi	Hindi	nplurals=2; plural=(n != 1);
hne	Chhattisgarhi	nplurals=2; plural=(n != 1);
hr	Croatian	nplurals=3; plural=(n%10==1 && n%100!=11 ? 0 : n%10>=2 && n%10<=4 && (n%100<10 || n%100>=20) ? 1 : 2);
hu	Hungarian	nplurals=2; plural=(n != 1);

Continued on next page

Table 2 – continued from previous page

ISO	English name	Plurals header in .po files
hy	Armenian	nplurals=2; plural=(n != 1);
I		
ia	Interlingua	nplurals=2; plural=(n != 1);
id	Indonesian	nplurals=1; plural=0;
is	Icelandic	nplurals=2; plural=(n%10!=1 || n%100==11);
it	Italian	nplurals=2; plural=(n != 1);
J		
ja	Japanese	nplurals=1; plural=0;
jbo	Lojban	nplurals=1; plural=0;
jv	Javanese	nplurals=2; plural=(n != 0);
K		
ka	Georgian	nplurals=1; plural=0;
kk	Kazakh	nplurals=2; plural=(n != 1);
kl	Greenlandic	nplurals=2; plural=(n != 1);
km	Khmer	nplurals=1; plural=0;
kn	Kannada	nplurals=2; plural=(n != 1);
ko	Korean	nplurals=1; plural=0;
ku	Kurdish	nplurals=2; plural=(n != 1);
kw	Cornish	nplurals=4; plural=(n==1) ? 0 : (n==2) ? 1 : (n == 3) ? 2 : 3;
ky	Kyrgyz	nplurals=2; plural=(n != 1);
L		
lb	Letzeburgesch	nplurals=2; plural=(n != 1);
ln	Lingala	nplurals=2; plural=(n > 1);
lo	Lao	nplurals=1; plural=0;
lt	Lithuanian	nplurals=3; plural=(n%10==1 && n%100!=11 ? 0 : n%10>=2 && (n%100<10 || n%100>=20) ? 1 : 2);
lv	Latvian	nplurals=3; plural=(n%10==1 && n%100!=11 ? 0 : n != 0 ? 1 : 2);
M		
mai	Maithili	nplurals=2; plural=(n != 1);
me	Montenegro	nplurals=3; plural=n%10==1 && n%100!=11 ? 0 : n%10>=2 && n%10<=4 && (n%100<10 || n%100>=20) ? 1 : 2;
mfe	Mauritian Creole	nplurals=2; plural=(n > 1);
mg	Malagasy	nplurals=2; plural=(n > 1);
mi	Maori	nplurals=2; plural=(n > 1);
mk	Macedonian	nplurals=2; plural= n==1 || n%10==1 ? 0 : 1; *Can’t be correct needs a 2 somewhere*
ml	Malayalam	nplurals=2; plural=(n != 1);
mn	Mongolian	nplurals=2; plural=(n != 1);
mni	Manipuri	nplurals=2; plural=(n != 1);
mnk	Mandinka	nplurals=3; plural=(n==0 ? 0 : n==1 ? 1 : 2);
mr	Marathi	nplurals=2; plural=(n != 1);
ms	Malay	nplurals=1; plural=0;
mt	Maltese	nplurals=4; plural=(n==1 ? 0 : n==0 || (n%100>1 && n%100<11) ? 1 : (n%100>10 && n%100<20) ? 2 : 3);
my	Burmese	nplurals=1; plural=0;
N		
nah	Nahuatl	nplurals=2; plural=(n != 1);
nap	Neapolitan	nplurals=2; plural=(n != 1);
nb	Norwegian Bokmal	nplurals=2; plural=(n != 1);
ne	Nepali	nplurals=2; plural=(n != 1);

Continued on next page

Table 2 – continued from previous page

ISO	English name	Plurals header in .po files
nl	Dutch	nplurals=2; plural=(n != 1);
nn	Norwegian Nynorsk	nplurals=2; plural=(n != 1);
no	Norwegian (old code)	nplurals=2; plural=(n != 1);
nso	Northern Sotho	nplurals=2; plural=(n != 1);
O		
oc	Occitan	nplurals=2; plural=(n > 1);
or	Oriya	nplurals=2; plural=(n != 1);
P		
pa	Punjabi	nplurals=2; plural=(n != 1);
pap	Papiamento	nplurals=2; plural=(n != 1);
pl	Polish	nplurals=3; plural=(n==1 ? 0 : n%10>=2 && n%10<=4 && (n%100<10 || n%100>=20) ? 1 : 2);
pms	Piemontese	nplurals=2; plural=(n != 1);
ps	Pashto	nplurals=2; plural=(n != 1);
pt	Portuguese	nplurals=2; plural=(n != 1);
pt_BR	Brazilian Portuguese	nplurals=2; plural=(n > 1);
R		
rm	Romansh	nplurals=2; plural=(n != 1);
ro	Romanian	nplurals=3; plural=(n==1 ? 0 : (n==0 || (n%100 > 0 && n%100 < 20)) ? 1 : 2);
ru	Russian	nplurals=3; plural=(n%10==1 && n%100!=11 ? 0 : n%10>=2 && n%10<=4 && (n%100<10 || n%100>=20) ? 1 : 2);
rw	Kinyarwanda	nplurals=2; plural=(n != 1);
S		
sah	Yakut	nplurals=1; plural=0;
sat	Santali	nplurals=2; plural=(n != 1);
sco	Scots	nplurals=2; plural=(n != 1);
sd	Sindhi	nplurals=2; plural=(n != 1);
se	Northern Sami	nplurals=2; plural=(n != 1);
si	Sinhala	nplurals=2; plural=(n != 1);
sk	Slovak	nplurals=3; plural=(n==1) ? 0 : (n>=2 && n<=4) ? 1 : 2;
sl	Slovenian	nplurals=4; plural=(n%100==1 ? 0 : n%100==2 ? 1 : n%100==3 || n%100==4 ? 2 : 3);
so	Somali	nplurals=2; plural=(n != 1);
son	Songhay	nplurals=2; plural=(n != 1);
sq	Albanian	nplurals=2; plural=(n != 1);
sr	Serbian	nplurals=3; plural=(n%10==1 && n%100!=11 ? 0 : n%10>=2 && n%10<=4 && (n%100<10 || n%100>=20) ? 1 : 2);
su	Sundanese	nplurals=1; plural=0;
sv	Swedish	nplurals=2; plural=(n != 1);
sw	Swahili	nplurals=2; plural=(n != 1);
T		
ta	Tamil	nplurals=2; plural=(n != 1);
te	Telugu	nplurals=2; plural=(n != 1);
tg	Tajik	nplurals=2; plural=(n > 1);
th	Thai	nplurals=1; plural=0;
ti	Tigrinya	nplurals=2; plural=(n > 1);
tk	Turkmen	nplurals=2; plural=(n != 1);

Continued on next page

Table 2 – continued from previous page

ISO	English name	Plurals header in .po files
tr	Turkish	nplurals=2; plural=(n > 1);
tt	Tatar	nplurals=1; plural=0;
U		
ug	Uyghur	nplurals=1; plural=0;
uk	Ukrainian	nplurals=3; plural=(n%10==1 && n%100!=11 ? 0 : n%10>=2 && n%10<=4 && (n%100<10 || n%100>=20) ? 1 : 2);
ur	Urdu	nplurals=2; plural=(n != 1);
uz	Uzbek	nplurals=2; plural=(n > 1);
V		
vi	Vietnamese	nplurals=1; plural=0;
W		
wa	Walloon	nplurals=2; plural=(n > 1);
wo	Wolof	nplurals=1; plural=0;
Y		
yo	Yoruba	nplurals=2; plural=(n != 1);
Z		
zh	Chinese[2]	nplurals=1; plural=0;
zh	Chinese[3]	nplurals=2; plural=(n > 1);

3.4 Valid Accelerators

Accelerators are used to allow quick access to items in menus and dialogs. Usually the user accesses them by pressing `Alt+X` where `X` is a character found on their keyboard.

In certain locales a keyboard can accept various characters including those with diacritics, therefore it is important to gather data for those locales so that the pofilter accelerator check can work well for those locales.

This mechanism could also be used in locales with multiple keyboards to ensure that only characters that appear on all keyboards are used in the locale in which case you might specify a list that is shorter then the normal English.

Note: Please enter the full set of characters for your language including lower and capital letters. Start with common Latin characters and add your specific diacritics at the end. Please provide a reason so that future language users know why you have chosen the given set if needed.

Language	Characters	Reason
English	abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ 1234567890	
Finish	abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ 1234567890 äöÄÖ	Bug 289
Polish	abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ 1234567890 ąćęłńóśźżĄĆĘŁŃÓŚŹŻ |	

[1] http://wiki.arabeyes.org/Plural_Forms
[2] zh means all districts and all variants of Chinese, such as zh_CN, zh_HK, zh_TW and so on.
[3] In rare cases where plural form introduces difference in personal pronoun (such as her vs. they, we vs. I), the plural form is different.

3.5 Resources

- ISO639 – use this to determine your language’s ISO code.
- FOLDOC – The Free On-line Dictionary of Computing – meanings of technical terms.
- If FOLDOC doesn’t have it, try Wikipedia, The Webopedia Computing Dictionary, or Google (try searching for “TERM definition” or “TERM meaning”)
- High-Tech Buzzwords will tell you the meaning of just about any computing or technical acronym! Look up those annoying and cryptic names like GPSD, AARNET and MTA.
- The alphabetic file extension list tells you what those file-extension abbreviations like .csv and .php really mean!
- dict.org – various online dictionaries that you can use to help define English words and thus help you create or find equivalents in your language.
- Wiktionary is a wiki-based dictionary system available in many languages, to which anyone can contribute. You can use its current resources, and build your own glossaries there.
- *Language names* – how do *you* write the name of your language?
- *Plural forms* – what plural forms does your language use?
- *Display settings* – what settings are necessary for ideal display of your language?
- Quotation Marks – what does *your* language use for quotation marks? Also how to set language-specific quotes in CSS. Thanks to Jutta Wrage for creating and maintaining this page.
- babelzilla – List of English – <language> – glossaries of terms used in the mozilla extensions
- Open-Tran – site that offers access to the translations of open source projects, as a kind of central translation memory

3.5.1 XXX

Language-specific resources

Vietnamese — Ting Vit

- The most common Net acronyms in Vietnamese lists many online, technical or chat abbreviations in both English and Vietnamese.
- Lac Vit English-Vietnamese dictionaries, including their computing dictionary, which is probably the most current computing dictionary available online, although the whole dictionary site is often unavailable
- KSVN English-Vietnamese, Vietnamese-English dictionaries which seem to have some words and phrases you don’t find elsewhere: their databases are definitely more current than some
- The Free Vietnamese Dictionary Project has a really impressive array of Vietnamese dictionaries to and from various languages, and is constantly improving it. This is probably the most thorough and reliable database of Vietnamese dictionaries. They are also available to run offline, on desktop/laptop computers or PDA.
- The English-Vietnamese Glossary Service is an excellent initiative, though currently hampered by lack of volunteer time. It should develop into an authoritative resource.

Dutch — Nederlands

- English-Dutch dictionary of the Dutch Translation Project team contains English to Dutch translations of common terms in open-source software that the translators' community has agreed upon. Their section for translators also contains links to other useful resources.

African languages

Several African languages gather localisation terminology in glossmaster: http://www.it46.se/glossmaster/

Other languages

In Trasno project wiki you have a list of glossaries for several languages.

Your language

- Insert your glossary, dictionary or acronym page here! :)

3.6 FOSS L10n Calendar

This calendar was created as a tool to all FOSS localisers and contains any and all significant events such as string- and code-freeze dates, etc.

Please contact Walter (walter translate org za) if you would like to help managing this calendar.

- Calendar sources: HTML, iCal, XML
- Currently tracked projects: Gnome, KDE, Mozilla Firefox, Mozilla Thunderbird, Ubuntu Intrepid Ibex

3.6.1 Other

There is also a Translate.org.za events calendar available here.