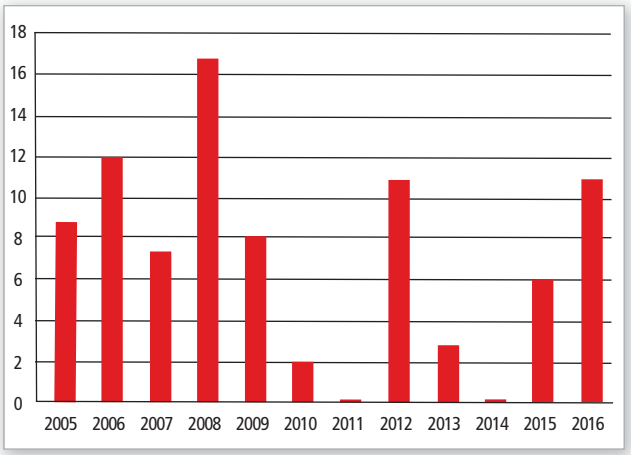
SEE TEXT ONLY

Improving the FreeBSD Translation Tools

By Warren Block

The first question that comes up about translation is "Why bother?" Or maybe, to put it more politely, is the effort of translating worth the result? A large portion of the world is inhabited by people who could benefit from using FreeBSD—and can't—because they don't speak English even as a second or third language. There is a huge benefit that we can bring to them.



Year	New translations
2005	9
2006	12
2007	7
2008	17
2009	8
2010	2
2011	0
2012	11
2013	3
2014	0
2015	6
2016	11

Secondly, translators are usually the ambassadors into a new area, region, or country. They are not just translators; they use whatever they have translated. If you can make the job easier for translators, you can also increase the number of people who are using your system in those areas.

Finally, you can enlarge the community. Metcalfe's Law, roughly paraphrased, states that the value of a telecommunications network is proportional to the number of nodes in that network. In other words, the more nodes, the more valuable it is. With the Internet, of course, if you had only two nodes on it, it would not be that useful. But if you have a million, it's much more useful. And I say that FreeBSD is a telecommunications network, and people are the nodes in that network. More users mean more people discovering bugs, submitting patches, adding features, enhancing documentation, and all this increases the value of FreeBSD for the very same people who are doing that work. It's a feedback loop. That is the value of translation.

The FreeBSD Documentation

There are several different categories of FreeBSD documentation. There are the books and articles, which are

marked up in DocBook XML. That is not just the famous *FreeBSD Handbook*, but also other books like the *Porter's Handbook*, which describes how to port programs to FreeBSD. It's a large, detailed, continuously updated book that is well worth a read. There are also a number of stand-alone articles marked up with DocBook XML. Using DocBook XML gives us the ability to render that source into a number of output formats like HTML, PDF, ePUB, PostScript, or even plain ASCII.

Secondly, we have our man pages. Ten or twenty years ago, some of the open-source world didn't quite see the advantage of man pages, or didn't understand it, and kind of took a wrong turn. But for the BSD community, this is a major feature. A man page is not a tutorial; it's meant to be a quick reference for when the exact format of a command or some configuration detail can't be remembered. You can look in the man page and there it is. We have a huge resource in those man pages, and they are continuously updated and improved. They use the mdoc(7) markup language, mostly, and there are some old ones that use roff or troff.

There is other documentation. Part of the FreeBSD Documentation Project's domain includes the source. If there is an error in spelling or grammar or clarity in source files, we are allowed to edit those. This also holds for text files, or any file, really. That is the scope of the FreeBSD Documentation Project.

Whitespace

One surprisingly difficult problem we encounter with translation is whitespace. These are the blank areas between text: spaces, tabs, line feeds, carriage returns, and a number of others. Being invisible, these characters are hard to see, and this makes them difficult to explain. The reason they matter in translation is that when a document is edited, translators have no easy way to tell if the content changed and a new translation is required, or if only whitespace changed and the existing translation is still valid. A typical example is when someone rewraps a paragraph to fit a particular line-length in an editor. The content is still the same, but that is difficult for translators to see without rereading the whole thing. At worst, the translator might retranslate the document unnecessarily. That is very demotivating and a serious problem when we depend on volunteer translators.

Whitespace changes have traditionally been managed in the same way as is usually done with source code. Changes to content are done in one commit; then a second commit changes whitespace—and only whitespace—to modify formatting. Whitespace-only changes to documents usually have a commit message that points out that content has not changed and the document does not need any new translation.

Separating content and whitespace commits helps translators, but adds a serious amount of work for people editing the original documents. Only content or whitespace can be changed at a single time, so documents become ragged and poorly formatted. When working on the follow-up, whitespace-only change, errors in content are often found. These cannot be fixed immediately, but require another round of content change and then yet another whitespace-only change. So, whitespace is a problem for pretty much everyone working on documentation.

The Old Translation Method

The traditional FreeBSD document translation method is simple in concept: translators work by commits. Translators work through the changes made to the English documents, translating those changes into the target language.

In practice, this method is difficult and time-consuming for translators. It is fully manual, giving no assistance to the translator. Quite the opposite, in fact: translators must work through commits in order, essentially having no control over the size of a change to the original document. A single sentence might have changed, or a whole chapter rewritten. Until that change has been translated, subsequent changes cannot be translated, and this can prevent other translators from working on the document. The scale of a change is also a problem with volunteer translators, who might see a large change as too much work for the time they are willing to contribute.

There is no formal method, but translation teams typically keep track of the latest translated version of a document with a comment noting the last commit number translated from the English document. This, too, is completely manual.

To locate and work by commits requires a non-trivial familiarity with the version control system. Because translators are working directly with DocBook XML source files, a non-trivial familiarity with DocBook is also often required. The quality and quantity of translations created with this old method is a testament to the dedication of the translation teams.

After all this overhead, the translator has finally located the next change to the English document

that must be added to the translated version. This change could be anything: new text added, old text removed, or both. At some point, the translator will compare the previous translated document with the diff file containing the changes. Because this method is fully manual, there is no assistance in locating the place in the document source where the changes occurred. Translations do not match the English documents line-for-line, so the translator is forced to locate the place where changes must be made. This can end up taking a lot of time when the change to the English document was non-trivial.

Finally, the translator can get to the one thing they wanted to do: creating a translation of the English document in the target language. Prospective translators do not respond well to this demotivating, labor-intensive workflow.

The New Translation Method

Something had to change. In 2012, Thomas Abthorpe and Benedict Reuschling were talking about a new translation method using "PO files." Even though I am a monolingual American, this work was obviously very important, and I slowly learned about ways to use this new translation method. In 2015, we finally had a working system.

The new translation method uses gettext, a program developed so vendors would not have to recompile programs so that their messages could be in a different language. Instead, a Portable Object (PO) file was created that contained both the English strings and their translated equivalents. After the user defined their locale, the strings were shown in the local language.

At first glance, this does not seem like a system that would lend itself to translating entire documents, but it does turn out to work surprisingly well for that. A program is used to extract strings from the original English DocBook XML source into a PO file. Translators use a PO editor to edit these files. The English string is shown, and the translator enters the equivalent translated string next to it. There is a 1:1 correspondence between the original and the translation. The translator does not have to look through source files to determine where the changes must be made. Whitespace goes from being a serious problem to mostly not being an issue. Finally, there is a "translation memory" that offers to reuse translations known from earlier uses. This is neither as good nor as bad as people tend to imagine, but it can help reduce the translator's workload by perhaps 5–15%.

The new translation method has only three steps:

1. Run 'make po' to extract translatable strings from the English document. If a translation already exists, it is preserved and updated with strings that have changed.

2. Run a PO editor and enter translations next to the English strings.

3. Run 'make tran' to build the translated version of the original document.

That is the entire process, all of it. The translator does not need extensive knowledge of the version control system or DocBook markup. They do not have to translate a particular change or entire large changes. Instead, they can do as much or as little translation work as they want without impeding other translators. Automation assists with the translation where possible, and PO editors typically show how much of the document has been translated. It is a huge change from the old translation method and lets volunteer translators actually contribute rather than driving them away.

Implementation

There are several programs that can extract translatable strings from XML files. The one chosen for this project was itstool (http://itstool.org), because of its simplicity and standards compliance. Some small utilities from the gettext-tools package are also used. itstool requires Python, which is already a required dependency of the documentation port (textproc/docproj). The gettext-tools port (devel/gettext-tools) is also likely already installed on a system used by a documentation writer or translator, so the overhead of these tools is very small.

Outcome

The new PO translation system went live at the end of August 2015. In the remainder of 2015, there were six newly-created translations of articles and books into German, Dutch, Spanish, Chinese, and Korean. This compares favorably to the numbers for 2013, when there were only three new translations, and 2014, when there were no new translations at all. In 2016, there

have been another 10 new translations, including two very large translations of the *Handbook* and *Porter's Handbook* into traditional Chinese. These numbers admittedly disregard the size of new translations and ongoing maintenance on existing translations. However, they demonstrate that simplified translation methods can be an enabling technology, and that given better tools, volunteers are capable of producing large quantities of useful work.

Challenges

We do still have some challenges, and I call these "challenges" because when you have a problem, it's just a problem. When you have a challenge, you are challenged to come up with a solution.

We have things that should not be translated. A prime example is our article that contains developer PGP encryption keys. That is all it contains, two sentences of introduction (essentially, "Our developers have PGP keys. Here they are.") and then 600 pages of PGP keys. These are just numbers. We don't want these translated, because the translated form of them is identical to the original. It wastes the translator's time by even showing them these strings. But we also want a single source for this type of information. If these strings were translated, they would be copied into the translated file, making multiple copies of them and guaranteeing that some will always be out of date in the translated version. So, we need a way to mark up the source to say "this string should not be translated." Ideally, the translator would never even see that string. Instead of seeing 600 pages of XML source, they would only see the two sentences of introduction.

Searches for standard ways of marking strings that should not be translated did not result in any obvious way of accomplishing that. There were a few organizations that had used toolchain-specific methods that appeared to not be applicable to our documentation. A messy hack of using XML processing instructions might be workable with our toolchain, but a better way is still preferable. The search is still on, and suggestions are always welcome.

Another challenge is preserving the huge amount of work put into translations that were created with the old translation method. Some of

RootBSD
Premier VPS Hosting
RootBSD has multiple datacenter locations,
and offers friendly, knowledgeable support staff.
Starting at just $20/mo you are granted access to the latest
FreeBSD, full Root Access, and Private Cloud options.
www.rootbsd.net

these are very current, and we want to preserve as much of that work as possible. Ideally, we would convert those translations into PO translations without losing any of that work. The GNOME xml2po program can take an English original document and a fully translated version and produce a PO file from them. However, the two documents must correspond line to line exactly. But our translation teams have their own rules for line wrapping, so the original and translation do not match. There might be other programs out there to do this based on matching the XML elements of the two files. It is likely that not much effort has been put into conversion programs because conversion for most organizations is a one-time occurrence.

Documentation translators would benefit from seeing a few lines of text from before and after the source string to get an idea of the context in which it is used. Many PO editors do not show much context. The original use of gettext was for translating program prompts where there was little or no context to show. With documentation, that is very different, and the job of translating is easier if surrounding context is shown. This is not a technical problem, because the PO files already have a comment with each line showing its line number from the original XML file.

There are many PO editors, but only a few have been ported to FreeBSD. Currently, there are three: editors/poedit, devel/gtranslator, and devel/lokalize. Having more PO editors in ports will make it easier for translators to choose an editor that is well-suited to translating the specific document or target language. A nearly functional port of the well-regarded Virtaal (http://virtaal.translatehouse.org/) is available and probably does not need much more effort to complete. Java-based PO editors are available, and the web-based online Pootle system is also in ports as textproc/pootle. PC-BSD had used Pootle for their translations. PC-BSD has now been renamed TrueOS, and currently uses Weblate (https://weblate.org/en/) for web-based translation. There are also commercially hosted sites with similar operation, like Transifex (https://www.transifex.com/), which is free for open-source use.

Potential

The most obvious and compelling possibility with PO-based translations is translation of the FreeBSD man pages. While itstool is strictly for XML files, the Debian po4a package is capable of extracting strings from man pages to PO files. A fully translated set of FreeBSD man pages would be incredibly valuable. With the barriers to translation lowered by the PO system, this becomes possible.

New translators often become FreeBSD documentation contributors and committers. PO translation offers an easy way to begin contributing and an introduction to the community which can lead to increased participation.

New translations of documentation will bring in new users from new regions and countries, offering their own unique perspective on problems and opportunities for the community. Some of those people will continue on to become contributors, making FreeBSD better for everyone who uses it.

What We Learned

The journey to make PO translation tools usable on FreeBSD taught us some related things.

The value of cross-pollinating with other projects should not be underestimated. Some of the most valuable insights were gained when Ryan Lortie of the GNOME project and I were talking before or after presentations at BSDCan. Different projects have different experiences, and two or more projects can benefit by sharing their knowledge. We should not look at other projects as competitors, but as rich gold mines of information.

The old translation method is essentially a commit-by-commit porting effort, porting an English document to another language. When the X11 team asked about commit-by-commit versus file-by-file porting, we were able to identify some of the problems inherent to the commit-by-commit method:

- commits must be done in order, even if a critically important later commit is needed
- usually the porter is forced to work in units of an entire source commit, regardless of how large it might have been, and the project can be stalled while one porter struggles with a large commit
- even though earlier commits can be entirely erased by later ones, the work to port them must still be done because later commits depend on their presence.

Technical debt can be a serious problem. When we tell people how we include chapters in XML files by defining them as entities, and they

pause and say, "We didn't think anyone was still doing it that way," it is a warning sign. Even if we follow standards, there is a danger that the new tool might not support the old methods. Ultimately, this could leave documentation or translation work stranded while new methods are implemented. Keeping up-to-date with industry standards is usually less painful if it is done gradually rather than as a sudden, urgent need.

Related to the previous point about technical debt: our documents need to be switched to UTF-8. It is long past time, and I have written a program to do this. All that remains is the hard work of verifying that the conversion works correctly.

Acknowledgments

The work of bringing the PO translation tools to the FreeBSD documentation has been under way for several years. Many people have contributed, both from within and outside the FreeBSD Project. This list is not complete, although anyone missing is due to forgetfulness on my part. My sincere thanks to everyone for their contribution to making FreeBSD more available to people around the world!

Thomas Abthorpe
Glen Barber
Mark Felder
Hiroki Sato
René Ladan
Ryan Lortie
Koop Mast
Shaun McCance
Chris Petrik
Benedict Reuschling
...and many others.

Links

Material discussed in this article, including the presentations, scripts used to collect statistics, graphs, and the Virtaal port, is here:

http://wonkity.com/~wblock/translation/

WARREN BLOCK has been using FreeBSD since 1998 for such diverse things as servers, documentation creation, network monitoring, and enterprise computing. He has been a FreeBSD documentation committer since 2011 and a member of the Documentation Engineering team since 2014. In 2016, he began working on the FreeNAS documentation for iXsystems Inc.

ServerU

Rack-mount networking server

Designed for BSD and Linux Systems
Up to **5.5Gbit/s** routing power!

Made for FreeBSD

PERFECT FOR

- BGP & OSPF routing
- Firewall & UTM Security Appliances
- Intrusion Detection & WAF
- CDN & Web Cache / Proxy
- E-mail Server & SMTP Filtering
- Anti-DDoS and clean pipe filtering

KEY FEATURES

- 6 NICs w/ Intel igb(4) driver w/ bypass
- Hand-picked server chipsets
- Netmap Ready (FreeBSD & pfSense)
- Up to 14 Gigabit expansion ports
- Up to 4x10GbE SFP+ expansion

Designed. Certified. Supported

1Gbit/s Copper	Ports	Chipset
L800-G808-1	8x Gbe RJ-45 ports	8x Intel i210 AT; PEX8618
L800-G808-2	8x Gbe RJ-45 ports	8x Intel i210 AT; PEX8618
L800-G428-1	4x Gbe RJ-45 ports	1x Intel i350 AM4
L800-G428-2	4x Gbe RJ-45 ports	1x Intel i350 AM4
1Gbit/s SFP (Fiber)	Ports	Chipset
L800-S406-1	4x Gbe SFP ports	i350-AM4
10GbE Copper	Ports	Chipset
L800-T202-1	2x 10Gbe RJ-45 ports	Intel X540
L800-T203-1	2x 10Gbe RJ-45 ports	Intel X540
10GbE SFP+ (Fiber)		Chipset
L800-X204-1	2x 10Gbe SFP+	Intel 82599ES
L800-X205-1	2x 10Gbe SFP+	Intel 82599ES
L800-X405-1	4x 10Gbe SFP+	Intel 82599ES; PEX8724

contactus@serveru.us | www.serveru.us | 8001 NW 64th St. Miami, LF 33166 | +1 (305) 421-9956