大陆简中自由软件本地化工作指南

Free Software Localization Guide for Chinese (China)

By Aron Xu, Mingye Wang, et al.

Version 1.5.2; Released under CC BY-NC-SA 3.0 Unported.

本文档提供自由软件本地化的一般性指引，主要包含 Gettext 翻译的编辑、验证和提交，以及目前较通行的写作风格信息和格式建议。

版本历史

1.5 (2016-01-17)　引入欠了两个月的斜体注释，工作重点转为向 zh_TW 移植。**Patch1** 03-11 添加 GitHub 上积压的一些关于单复数、对齐和标点的内容。**Patch2** 04-28 换行部分的小修正。

1.4 (2015-11-18)　导入繁体中文版本中做出的一些有益改动，包含符号格式、日期速查和 glib 日期；采纳 Dongsheng Song 的提示加入 Unicode General Punctuation 表链接。统一对译者的称呼为“你”。**Patch1** 11-21 加入 W3C《中文排版需求》和一些项目自身的跨语言排版方针。**Patch2** 11-29 不翻译列表增加英文缩写和单位名；增加已知的分支列表。**Patch3** 11-30 hyphen & dash。**Patch4** 12-13 澄清省略号和快捷键的前后次序及快捷键大小写。

1.3 (2015-10-30)　语言名称更新，GUI 使用 CJK 括号，添加 Translate Toolkit，拆开文件格式和写作格式要求章节。**Patch1** 11-09 收入 iKDE 用词方面的部分内容。**Patch2** 11-11 修正日期格式的空格问题并增加 POSIX.1 2013 链接；补充帮助消息描述；关于 CJK 字体中制表符的额外提醒。

1.2 (2015-09-12)　重制 ODT，重构文档结构。添加必要注解，删减陈旧内容，今年是 2015 年（发行注记、争吵）。

1.1 (2009-11-07)　更新格式信息。

1.0 (2009-08-26)　初始版本，原网页以 Google Docs 发布于 i18n-zh。

版权信息

目录

1 工作流程

自由软件翻译的工作流程基本分为准备、协调、翻译、提交四步，不同项目各有不同。

一些翻译平台，如 Launchpad，Transifex 和 Crowdin，提供基于网页的提交方式。这种平台在协作上的分块冲突相应较少，且常提供“翻译建议”暂存区等待处理。这种平台的原生数据格式常为 XLIFF。

1.1 准备

为保证翻译的质量，请在翻译前做好如下准备：

- 对目标软件基本熟悉，我们强烈不赞成翻译自己不熟悉的软件或文档。
- 详细阅读本文档和全部与工作相关和要翻译内容相关的文档，若有 TeXinfo 文档词语解释的话须熟读。
- 英文词典、项目术语表、术语参考资料和维基百科。这些是术语、缩写翻译的重要参考。维基百科一些术语翻译可能较为陈旧或生僻，此时可考虑 Google 此术语配合语言限制为简体中文来寻找常用的译法。特别注意不要翻译 Copyright 这个单词[1]，也不要展开翻译任何缩写（GNOME 建议），特别是众所周知的 SI 单位名。
- 源码包（心情好的话）。PO 文件内常会记载字符串在程序何处被使用，一些编辑器更能直接打开目标位置。

如果你使用网页翻译平台，现在就可以直接开始翻译了。否则的话，你需要获得翻译文件：

- 如果软件包文档里面有翻译指南的话，遵循那个指南的翻译步骤。
- 网页搜索“项目名 + 著名翻译平台名”，如“tor Transifex”，然后获取 po 文件，同时准备与相应平台上的团队协调任务分配。
- 解开源码包，寻找 PO/POT 文件或者其他类型的翻译文件。如果发现没有 PO 文件只有 POT，可以使用 `msginit` 工具创建新 PO 或手动复制再编辑。大部分 PO 编辑器也会处理这件事。

尽管我们的 PO 文件本身属于纯文本，但是使用专门的编辑器总是能让你的体验好上不少（例如避免手滑）。POEdit 和 Emacs 的 PO-Mode 都是不错的选择。如果你还有额外的需求（例如处理更多格式，包括使用 TBX 术语表、交换 TMX 翻译记忆），你该试试 OmegaT 或者小巧一点的 Virtaal。

1.2 协调

协调工作的目的是为了防止翻译上的冲突，也就是重复翻译的情况。这种情况不但造成重复劳动，还会造成合并困难，可谓事倍功半。

通常情况下的协调工作是通过向小组的邮件列表发送电子邮件说明你要负责的工作，若要翻译的软件或文档不存在则自动成为其协调人开始负责其翻译。若已经有人在进行此项翻译，则此后需要与之联系以商定分工，请在发送邮件到邮件列表时抄送给原负责人一份，一周内无回复则视为同意你的请求。开始翻译前若需要相关的帐号应提出申请。

1 https://www.gnu.org/licenses/gpl-howto.html: Always use the English word “Copyright”; by international convention, this is used worldwide, even for material in other languages.

GNOME 由于使用了 Damned Lies 站点，请直接使用站点功能进行保留锁定和提交。如果你看到有人在之前的评论区提交了翻译，应该从他们的版本继续工作。为了节省人力，一般习惯专攻开发分支或最新稳定分支，然后再转移这些翻译到其他分支[2]。一般习惯在某分支 String Freeze，即不添加新字符串后再彻底完善其翻译。KDE 也有字串冻结的情况，不过协作方式麻烦一些。

对于在线翻译，如 Launchpad、Transifex，暂时还没有进行协调工作的有效方式。从另一方面讲，在线翻译不少都看得到他人的翻译建议和进度，对协调的需求也相应降低了。

1.3 翻译

本章节描述 Gettext PO 文件的翻译方式。如果你使用别的翻译方式，可以考虑跳过。

GNU Gettext 的一些信息会在之后提到。作为准备工作的一部分，你需要备好 gettext 工具集；我们同时假定你使用类似 GNU/Linux 的命令环境。如果你要和各种 XML 打交道，你可能还需要 intltool。

注意 不要修改 PO 文件的 msgid 项，否则这条翻了也没用，合并的时候还可能被忽略。如果原文 msgid 真的有错，你可以向项目开发者报告。

进行翻译本身非常简单，用编辑器打开就好了。如果你使用纯文本编辑器，你可能需要阅读下一章的文件格式信息。

1.3.1 检查翻译格式

PO 文件翻译后请使用以下命令进行格式检查：

```
msgfmt --statistics -cv foo.po
```

此命令会当前目录下生成一个名为 `messages.mo` 的文件，此文件便是编译后的机读文件。将此文件复制到 `/usr/share/locale/语言名/LC_MESSAGES/消息域名.mo`，再运行 `locale-gen`，就可以替换目前的翻译了（具体的消息目录仍然取决于程序本身）。你也该看看接下来对于 `--check-acclerator` 的描述，

你的 PO 编辑器有可能已经帮你做了这件事，所以你也可以不管这个。

1.3.2 复审翻译

使用 PO 文件时，应直接复审 PO 文件，并建议编译为 MO 文件安装在相应程序的目录下进行实测；对于文档，构建文档后精读结果文档。

如果发现问题，回到翻译步骤修正问题。

1.3.3 格式处理

翻译工具互相之间可能在细节处存在不同的格式设定，例如 POedit 默认和 wine 项目要求行宽度为 79 列，而不少其他项目又干脆让你不要半路换行。还有不少人和编辑器为了对齐会把 msgid 和 msgstr 行的文字内容留空，接下来的几行再慢慢接上。

要解决问题，你可以使用 msgmerge，一个用来把翻译适应到模板上或者合并进新翻译的工具：

```
msgmerge -no-wrap -U foo.po foo.po   # 不要半路换行，和自己适应（也就是不去做适应）
```

2 如果你看到页面提示其他分支有 on-going activities，请运用 msgcat, msgattrib 和 msgmerge 知识予以合并迁移作为基础。

```
msgmerge -w 79 -U zh_CN.po drunk.pot # 最大宽度 79 列，自动换行，适应到 drunk.pot
```

1.4 提交

不同项目提交翻译的方式不完全相同，多数情况下可以通过翻译小组的邮件列表进行提交，如果你有相关项目的 DVCS 帐号则请直接提交。如果没有负责的小组也没有相关的帐号，请把你的翻译发送到对应项目的翻译者邮件列表（一般名称类似 xxx-translators/i18n/intl/l10n；若没有翻译者邮件列表则发送到文档邮件列表，一般名称类似为 xxx-doc；若还是没有则发送到开发者邮件列表，一般名称为 xxx-dev/devel/developer），或者通过填写一个 Bug 的方式进行提交。

一些项目可能要求你提交 patch，此时你应该使用 `diff -Nau old.po new.po` 的形式生成 patch，然后视情况手动修改头部的文件路径。

注意 在请别人代为提交到 TP 项目时 Last-translator 不会是你，因为 TP 要求这项需要是提交者的名字和邮件地址，你的信息将会出现在文件头注释域的版权行中。

提交后请根据情况再发送一封邮件来取消原来协调翻译时的占用声明。

2 PO 文件格式和相关要求

GNU Gettext 是 GNU 的本地化框架，主要内容是一个喂进去英文吐出来翻译文件中的对应值的一组 C 函数和处理翻译文件的诸多工具。[3]

2.1 基础信息

PO 文件主要是 msgid 和 msgstr 一一对应的序列，外加一些单复数情况。

```
1.  #: actionlog/templates/object_action_list.html:8
2.  #. TRANSLATORS: 翻译者们大家好，这是我给你的注释。#: 开头的都是程序源码来的。
3.  msgid "Action"
4.  msgstr "干活"
5.
6.  #  msgid now comes in plurals. 我是个普通的注释，这种由译者留下。
7.  #: foo/templates/bar.html:180
8.  msgid "{0} result"
9.  msgid_plural "{0} results"
10. msgstr[0] "{0} 个结果"
```

2.2 文件头

PO 文件必须有一个空 msgid 的对应值以存储文件头的数据，并且其中有多个必须项。同时，PO 文件头部经常以注释形式表示版权信息。以下是一个范例 PO 文件头。

PO 文件头的版权信息如下所示：

3 其实是读 https://www.gnu.org/software/gettext/manual/gettext.html#PO-Files 干脆。

```
# Simplified Chinese translation for 这啥.
# Copyright (C) 哪几年 这谁的.
# This file is distributed under the same license as the 这啥 package.
# Aron Xu <happyaron.xu@gmail.com>, 2011-2015.
# Mingye Wang <arthur200126@hotmail.com>, 2015.
```

不少翻译平台都会强制你填入这些信息。一般来说，第一行是一个大致的描述，第二、三行是版权信息，接下来是翻译者名单。这些信息基本需要手动处理。哪几年的内容一般而言会像 1997，2000-2013，2015 这样用逗号隔开每一项。一些项目不允许用年份范围，只能一个个列出。[4]

接下来是存储 gettext 元数据的空 msgid 文件头：

```
msgid ""
msgstr ""
"Project-Id-Version: 这啥 0.0.1\n"
"Report-Msgid-Bugs-To: http://滚键盘.cn/bugs.html\n"
"POT-Creation-Date: 2015-07-02 12:31+0000\n"
"PO-Revision-Date: 2015-08-26 23:17+0800\n"
"Last-Translator: Mingye Wang <arthur200126@hotmail.com>\n"
"Language-Team: Chinese (simplified) <i18n-zh@googlegroups.com>\n"
"Language: zh_CN\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"
"X-Poedit-Basepath: C:/MSYS/source/gcc-4.6.0/gcc\n"
"Plural-Forms: nplurals=1; plural=0;\n"
"X-Generator: Poedit 1.8.4\n"
```

聪明的你很快就会想到，这些东西打印出来之后有一股 HTTP 头味。事情的确就是这样：

- **Project-Id-Version:** 项目名和版本。
- **PO(T)-Revision-Date:** 表示 PO 和 POT 的修改日期。
- **Last-Translator:** 最后的翻译者，发现整出了新锅子就甩给他。大部分编辑器会自动填上，用文本编辑器的话只能手动了。一些翻译平台会检查这个值的电子邮件是否为发件人或在团队中，所以找人代为递交时可能需要填上他人的名字。
- **Language-Team:** 翻译小组名（一般为语言名）和电子邮件地址。新建的时候要手动填一次。
- **Language:** 语言代码，遵循 `语言代码[_国家地区][@特殊属性]` 的格式。
- **Content-Type:** MIME 类型；`charset` 信息需要符合实际情况。提交应使用 UTF-8 编码。
- **Plural-Forms:** 复数形式，中文的话用 `nplurals=1; plural=0;` 就好——也就是只有一种复数形式，使用的复数形式序号永远是 0。`plural` 其实是一个 C 风格的整数表达式，返回复数形式元素的下标号。[5]如果担心“他们”这种情况，那就改用 `nplurals=2; plural=(n!=1);`。单复数内容含义不严格一致时，翻译依复数原文为准。例如，遇 `Last year` 和 `%d years ago` 成对出现，译 `%d 年前`。

4 https://www.gnu.org/licenses/gpl-howto.html: For software with several releases over multiple years, it's okay to use a range (“2008-2010”) instead of listing individual years (“2008, 2009, 2010”) if and only if every year in the range, inclusive, really is a “copyrightable” year that would be listed individually; and *you* make an explicit statement in your documentation about this usage.

5 https://www.gnu.org/software/gettext/manual/html_node/Plural-forms.html 给出了一个很多种复数的语言的例子。

和很多其他地方一样，X- 开头的项目表示其他拓展的数据项。

2.3 字符转义和格式

在翻译过程中你会遇到不少特殊含义的字符。有些会被你的编辑器突出显示，但你还是得知道含义。

2.3.1 Gettext 字符转义

Gettext 本身的转义序列和 C 的类似，也就是 \t 为 TAB，\n 为 LF，\v 为竖直制表符，\\ 为单个反斜杠，\" 表示单个直双引号。例如 he\"\\tllo\" 这个片段就会得到 he"　　llo"。

当你在一个消息的首尾处看到 LF 的时候，你也该保证你的消息首尾有对应的 LF，保证换行之类的不会闯祸。除了这个就没有特别需要注意的了。

2.3.2 编程语言格式

PO 文件中常会标记出 xxx-format 的字符串，这说明这个字符串遵循 xxx 语言的格式化方式。基本上记住 glibc printf 的方式就够处理大部分内容了。如果你不巧翻译了一些奇怪的东西，例如 scheme-format，你可能需要换脑子看看格式化命令是不是有跳转功能（对于 Scheme 的确就是这么用的）。

Gettext 的 msgfmt 对于认为是 c-format 或类似的字符串有一个拓展，可以指派使用第几个参数，例如 $2%s 就是把字符串后第二个参数当作字符串填进来。这样做可以有效地修正语序，而不需要费劲调整副词顺序。[6]

```
# 例如在 KDE 中某处的 qt-format:
msgid "%1 articles match rule %2"
msgstr "匹配规则 %2 的文章有 %1 个"
# GNOME 中 c-format 则是这样:
msgid "%d articles match rule %d"
msgstr "匹配规则 %2$d 的文章有 %1$d 个"
```

任何一个参数的顺序进行了调整，则在这一句译文中所有参数都必须注明原文位置，否则无法通过格式检查。

2.3.3 XML 实体转义

一些项目，例如 GConf，会涉及到裸露的 XML 部分。如果你要表达 & < > 这三个字符，需要分别使用 & < > 指代。一般来说直接模仿 msgid 的样子就好了。

2.3.4 翻译中的保留字

Gtk+/GConf 中经常出现 TRUE 和 FALSE 这样的字符串，多见于以 .glade 为后缀名的文件中。不要翻译它们——程序找不到它们会出错。除此之外，各种 possible values 也不该翻译，顶多用括号指明中文意思。

2.3.5 快捷键

在 Qt 和 Gtk 中都存在快捷键定义，分别使用 &x 和 _x 的格式指定快捷键 x。使用 msgfmt 的 --check-accelerators 可以检查这些字符（检查 Gtk 时加 =_）。

6 https://www.gnu.org/software/gettext/manual/gettext.html#c_002dformat-Flag 直接网页搜索 1$ 就行。

如果翻译出的文字中不含有原有的快捷键字符，那么在空格后跟一个括号包含那个字符的大写形式；否则使用翻译结果中的原快捷键字符。快捷键使用西文括号且不前置空格，此括号标记置于字串中省略号之前。

KDE/Qt 样例：

```
msgid "C&lear..."
msgstr "清除(&L)..."

#  不是 Glimmer 编辑器(&G) —已经有一个 G 可以用了
msgid "&Glimmer Editor"
msgstr "&Glimmer 编辑器"'
```

Gnome/Gtk 样例:

```
#  顺便注意一下 GNOME HIG 要求的正确省略号。
msgid "_Setup…"
msgstr "设置(_S)…"

#  不是“现在读取 CDDB(_C)”
msgid "Get _CDDB Now"
msgstr "现在读取 _CDDB"

#. 以下情况的翻译有点特别。
#. 复制为“编辑”菜单的条目，只有这样写才能保证显示正确！
msgid "/_Edit"
msgstr "/编辑(_E)"

msgid "/Edit/C_opy"
msgstr "/编辑(E)/复制(_O)"
```

2.4 注释和字符串属性

Gettext PO 字符串注释还提供多种属性。

```
#  译者注释
#. 程序员留给译者的注释
#: 文件:行号，更多的文件..
#, 各种属性，又一个属性..
#| msgid 模糊匹配的老结果
msgid "blah"
msgstr "blah"

#  未在 pot 找到对应的老翻译
#~ msgid "bl"
#~ msgstr "ah"
```

请注意他人和开发者留下的注释经常非常重要。对于其中留下的行号信息等上下文，一定要仔细考虑。特

别是对于短词，直译的选择可能有很多，上下文此时就更重要。例如 Start，单独翻译成开始，但是在 Start、Pause、Resume 都作为需要“被”的词时也许可以采用，启动，暂停，恢复这样的翻译。如果拿不准一句话的含义，那更要参考注释和上下文语境了，必要时还可能需要实际代入软件确认。

2.4.1 翻译属性

模糊 (fuzzy) msgmerge 模糊匹配的结果，或者是翻译者因为不确定而手动留下的标记。模糊匹配有时比较准确，有时却谬之千里。带有模糊标记的字段不会被 msgfmt 编译入 mo 文件，所以需要去除标记。

foo-format 见编程语言格式章节。

2.4.2 已淘汰翻译

老版本的程序中总有一些信息会在新版本中淘汰掉，这些译文由 msgmerge 程序自动设置为以 `#~` 开头，并被置于整个 po 文件的后方，直接删除它们不会影响当前版本的翻译内容，但是因为这些字符串还可能在以后的版本中重新出现，届时 msgmerge 还可以使用这些译文来提供翻译建议，所以如果它们没有影响到你的工作建议不要删除它们。

3 语言和排版要求

语言基本要求基本上就只有这几点：

1. 准确表述原文含义。
2. 中文应该意思清晰且符合中文表达习惯；自己的翻译多读几遍总没什么坏处。
   a) 如果你从其他中文语区借助 OpenCC 等工具进行字词转换导入，请注意仔细检查语序和用词。一般而言，这些消息都该先标为 fuzzy 再审阅一次。
   b) 中英文在语序习惯上区别较大，例如英文常后置动词[7]和使用被动句。例如 File is being saved，按中文的习惯是“正在保存文件”。英文中分词等时态的使用也要在翻译中用“正（在）”、“已（经）”、“了”、“完成”等体现。
3. 原文如果表达不清晰，中文应该意译，并且应根据上下文和注释进行推断并填补相应的信息。
   a) 这种情况不能太多，上下文必须看明。
4. 对同样短语的翻译，前后必须一致。善用文本编辑器的搜索功能和 `msggrep`。
5. 有一些词汇不该被翻译，例如前面提到的程序关键词。

此外，一致的语言风格对于用户体验的一致性很重要[8]。不同大项目中“你”和“您”的偏好这种细节也会在细心的用户面前变得非常明显[9]。

3.1 标点符号

基本上，除了圆括号和省略号[10]保持不变外，都该改成符合汉语风格的全角标点。西文排印时一般在句读后

7 这个得改……

8 https://github.com/sparanoid/chinese-copywriting-guidelines 是汉语排版的主要指南之一；其余的可以参考后记。

9 Google 你 `zh_CN filetype:po site:l10n.gnome.org` 得到 265 个结果，换成“您”则有 653 条，可见有些项目内现在也不怎么统一。

10 破折号在上一版本中是考虑到“——”有时会显示为乱码方框不用；然而六年过去了。

加一空格[11]来保证间隙正常，换到中文中自然要去掉。标点的完整用法实际上应参照《现代汉语词典》或者《新华字典》附录的那张《标点符号用法》[12]。一般鼓励使用 Unicode 通用标点（General Punctuation）。

1. `,` 可能对应逗号或者顿号。如果你发现原文中逗号后面没跟空格，那有可能是程序代码格式例子，不该进行转换（参见第九条）。当然，原作者出错也不是不可能。
2. `.` 可能对应句号、分号或者逗号（分句）。
3. `-` 若为连字之用，可不译（前后均为 CJK 的话），或保留原样（译文中前后有至少一个西文的话）。有时你会注意到它被误用作连接号 en-dash – 的用途，那也要反应过来并用中文连接号–译。就像 `Ubuntu-Debian Distro Family`，翻译成“乌版图—大便发行版家族”（我后悔用 Debian 举例了）。
   a. CLI `-` 若前后均有空格，则该认为是短破折号 en-dash 之代用品，可用单个字符–对译。
   b. 类似地，`--`，即 em-dash 代用品（长连接号或破折号），译作完整的中文破折号——。这几种 dash 本身也按照对应代用品的方法翻译。这些所谓的“连接号”在 UI 中实际常作破折号之用，好在替换逻辑一致。
4. GUI 程序中的括号 `(`，除了快捷键标记 `(&X)` 之外都翻译成中文括号（。CLI 程序保留英文括号，原因是历史上一些中文字体（如 SimSun）的圆括号很难看[13]，而 CLI 又是老字体遗留问题的博物馆。西文括号左括号左边、右括号右边应加空格，否则显得过于拥挤。快捷键标记不要加空格。
   a. 括号内容全为西文字符时，如 `(px)`，保持西文括号似更为美观，特别是后跟冒号 `:` 或内容全为小写时。
5. `"aaa"` 在 GUI 程序中一般应使用弯双引号 “a‘a’a”[14]，对 CLI 程序则应为直引号 `"a'a'a"`[15]。对于 `` `aaa' ``[16]除非是 m4 代码否则按照前面的规则走。对 `'aaa'` 同理。若原本就是弯引号，则不改变弯直，只按照简体中文用法双单交替。
6. `Foo:` 应翻译为 `Foo：`，而作为分隔符使用的 23:59 仍然应使用半角冒号。
7. `...` 省略号应保持不变，因为历史上的翻译者不便验证一个消息的上下文是菜单（菜单传统用 `...`）还是普通文字（可以用 …）。较新的一些程序原文已直接使用 … 省略号，保持不变照做就好。
8. `%q` 标记表示以可重用的方式输出文字，再乱加引号就搞砸啦！
9. 遇到说明英文标点的消息时，除了翻译为中文外，请在译文中加入该标点，并在标点前后加上弯（GUI）或直（CLI）引号。例如 separated by comma 译为“以半角冒号‘:’[17]隔开”，或“以半角冒号 ":" 隔开”。
10. 遇到说明菜单的消息时，例如：`System > Administration`，可以见到首字母大写，这样做除了模仿菜单本身首字母大写外，还能在同是小写的句子中吸引读者的目光。因此请使用半角方括号括住并且

11 句号也有加两个空格的，不过 en_US 里面基本看不到。

12 http://www.moe.gov.cn/ewebeditor/uploadfile/2015/01/13/20150113091548267.pdf / http://people.ubuntu.com/~happyaron/l10n/GB(T)15834-2011.html / http://www.zdic.net/appendix/f3.htm

13 当然 2015 年各种 Linux 平台上的字体选择都没那么难看（例如思源黑体和 Ubuntu），所以你可以考虑把现有的翻译都查找替换一遍。

14 实际上开源软件出现直引号在那里挂着就要小心是代码了，特别是双引号。所以要小心。

15 大陆简中和西文的弯引号占同一码位，在等宽字体中就成了个坑，因为两边对这个字符宽度的普遍认知分别是一列和二列。通过遵循西文和国际习惯在引号开闭处如括号一般加空格可以使得单列时显示不丑，但这样双列又会空间太大。省略号、破折号还有制表符也是这样。宽度不同同时也会造成其他问题，在一些终端下的文本编辑器中尤其明显。

16 `aaa' 在没有弯引号的打字机时代常被作为 ‘aaa’ 的替代品，今天还在小字符集的地方用。

17 这个单引号是因为中文引号嵌套才用的，实际翻译还是该怎么双引号就怎么来。

用箭头“→”（GUI）或大于号“>”（CLI）连接，如 [系统] > [管理]，括号与 > 之间需添加半角空格隔开。

不少项目另有自己的排版要求，例如 GNOME 的 HIG 和 Inkscape。如果你在源码中注意到乘号×而不是 x、原文用弯引号之类的情况，你也该认真注意自己所用的那个符号的正确性。

3.1.1 空格

在汉字（不是字符！）和英文字符、阿拉伯数字之间应加入空格，除非你翻译的是 LibreOffice 文档这种会自动处理空格的东西。上面提到括号等少数使用西文字符的情况遵循英文空格习惯。按键指南等语义中间有明显区别却没有引号的区域两端也要留出空格，例如通过 <tt>Alt</tt>+拖动 来调整。斜体（Italics / Oblique）的文字后若不是标点符号或空格，也应加入一个空格[18]。

包含 XML/HTML 标签的条目，如要在标签中的内容两侧添加空格，请把空格置于标签外侧，否则空格可能不会显示：这是 HTML 的语法手册。

3.2 换行、行宽和对齐

Gettext PO 文件本身对换行不敏感，以下这几种形式都是一样的：

```
msgid ""
"wut?"
msgstr ""
"蛤？"

msgid "wut?"
msgstr "蛤？"

# 因为会拼接，其实连这样都行：
msgid "wu"
"t?"
```

不过 Unix 终端机世界中存在一行 80 列宽的传统。再去掉 diff 占用的一行，就得到了 79 列的常见标准。

既然 CLI 下进行文档编辑时需要换行，那么 CLI 下的程序翻译也需要相应换行。将 79 除以每汉字二列宽的数值，就可以知道大致每行 39 个汉字就应用 \n 换行。在翻译 GUI 程序时，如果看到了 \n 且发现行尾长度相近，也可能是提示需要手动考虑行宽度进行换行。如果使用普通文本编辑器，在使用 \n 换行时建议在 PO 代码相应位置换行以保证可读性。旧版本的本文件还提示在 HTML 中做类似处理，然而实际上应该和 GUI 情况类似——一般不大推荐
 强行换行。

CLI 下还需要注意对齐情况。一般而言，TAB 是最稳妥的对齐方式[19]。有些程序（例如 minicom）可能会被 TAB 搞出计算错误，此时就必须手工使用空格对齐。手工对齐时，要保证自己的编辑器字体等宽且满足每汉字二列的习惯数值（插入广告：可以试试 Inziu Iosevka SC）。

永远不要在 GUI 程序中尝试增加原文没有的对齐，尤其是不要在中文词当中加空格（如 啊　啊）。如果真的想要对齐的话，那就尝试调整用词达成一致（例）。如果要用空格的话，那么只有宽度严格已知的空格有

18 中文原则上不应使用斜体，应优先用其他格式代替。

19 Mingye：我发现我被字体惯坏了，平常也习惯直接空格了。当然也还有 POEdit 只会显示个 \t 的错。

意义，如全角空格（U+3000）。

3.3 时间格式

鉴于关于日期和时间的译法十分复杂，现在将国家标准[20]中有关规定简要介绍一下，并给出推荐译法。以下是最需要注意的几个要点：

(1) 目前的 zh_CN 语言区域定义文件使用的日期格式内数字和年月之间不加空格，可使用 `LC_ALL=zh_CN.UTF-8 date +%c` 确认。

(2) 对于日期，必须按照年月日的顺序；年份一般用四位数字，而月、日必须使用带前导零的两位数字；必须使用“-”作为分隔符，或完全不使用分隔符，如 2004-01-03 或 20040103。

(3) 对于日期，时、分、秒都必须使用两位数字，若有分隔符只能为半角冒号，并使用 24 小时制。

(4) 同时表示日期时间时，应先写日期再写时间。在日期和时间都不使用分隔符的情况下，在中间添加字母 T 进行分隔，如 19850412T101530。

请注意以上 2~4 内容仅是 ISO 国际标准的一致交换需求，实际情况还应该按照语区具体习惯决定。时间的表示方法一般遵守 strftime(3)，基本就是 date(1) 的表示方法，可以用 man 很快查到。glib 另有一些拓展，可见于其文档中。

下表提供常见原文日期格式的中文对应，可置于翻译记忆中自动查阅：

原文	译文
%a	%A
%A，%B %e	%b%e日%A
%a %b %e	%b%e日（%a）
%A，%B %d	%m月%d日%A
%A，%B %e，%Y	%Y年%b%e日%A
%a，%d %B %Y	%Y年%m月%d日（%a）
%A，%B %d，%Y	%Y年%m月%d日%A
%d %B %Y	%Y年%m月%d日
%l:%M:%S %p	%p%l:%M:%S
%R:%S	%R:%S

3.4 命令行程序帮助消息

命令行程序中帮助消息的例子词，如 NAME 一般都应进行翻译。在接下来的文字中使用 某某 <名字> 某某 来进行指代；对于老翻译可以将 =FILENAME 替换为 =文件名 [21]，然后再对 FILENAME 按照两边加空格、一边加和不加三种情况替换为 <文件名> 以便快速操作。要特别小心帮助消息的对齐。

20 GB/T 7408:2005 / ISO 8601:2000. Old: http://people.ubuntu.com/~happyaron/l10n/GB(T)7408-94.pdf

21 这里有两个空格，于是对齐就补回来了。前面大小于号左右各一个空格是为了方便辨识。

3.5 其他细节

以下最早是 csslayer 对于一些用词的建议：

- -er 结尾名词：英语常直接转换动词为名词，可考虑翻译为 X 器，如 cleaner 翻译成“清理器”。
- 有时可省略 you：you 并非必须在句子中翻译成“你”。对于 Do you want to 之类的问题，直接译为“是否”显然更佳。
- 褒贬：有时一些词翻译时还要考虑褒贬，例如 KDE 中的 fancy 若翻译成“花哨”就变了味[22]，该考虑语境译为“华丽”。又如常用来形容老设备的 legacy，译为“传统”相比“老”更加含蓄。

4 工具使用速查

Gettext 是你几乎不可能避免使用的工具，Inittools 又是 Gnome 翻译的重头戏。除了这些之外，你也应该知道 Translate Toolkit 的几个黑科技工具（例如 pofilter 查错），以及不少 pot 是用 po4a 转换来的这件事。

4.1 Gettext

通用参数	`<blah> -o 输出文件名 [-w 行宽度|--no-wrap]`
生成机读	`msgfmt --statistics -cv filename.po --check-accelerators[=CHAR] POFILE...`
反转机读	`msgunfmt MOFILE...`
转移新旧	`msgmerge [--previous] [-C 翻译库.po] [-NU] 旧翻译.po 新模板.pot`
转换编码	`msgconv -t UTF-8 old_crappy_big5_or_gb.po`
新建翻译	`msginit -l zh_CN example.pot`
消息去重	`msguniq FILES...`
消息搜索	`msggrep`

4.2 Intltool

intltool 工具集实际上是一组脚本，用以实现一些常规的翻译文件维护和与目标 XML 文件的交互。此软件包一共含五个命令，对于翻译者来说比较常用的是 `intltool-update` 和 `intltool-merge`。这组工具多数需要在完整的源代码树中的 `po/` 子目录下运行。要了解更多关于 intltool 工具的信息，请阅读其手册页。

4.2.1 intltool-update：更新 POT 并与 PO 合并

通常使用它生成 POT 文件时使用 `intltool-update -p` 生成一个 POT 文件。

要使用它更新原有的 PO 文件，可以运行 `intltool-update zh_CN`。这样将会自动生成新的 POT 文件并更新 `zh_CN.po`，最后得到的文件是更新后的 `zh_CN.po`，也可以使用 `-o` 选项来定义输出到指定文件而非更新原来的 PO 文件。

还可以使用它来查看源代码中的 `POTFILES` 文件是否被正确维护，通常翻译者不需要做这项工作：

22 况且还有很多人嫌 KDE 太花哨，这是后话。

`intltool-update -m`。如果输出为空则代表一切正常，否则将有详细提示。

4.2.2 intltool-merge：交叉合并 PO 和 Gtk XML

```
intltool-merge --utf8 PO_DIR INPUT OUTPUT
```

其作用为将指定的 PO_DIR 目录中的所有 PO 文件同 INPUT 文件合并，并将输出写入到 OUTPUT 文件，OUTPUT 文件中包含了 INPUT 文件中的原始字符串和其他 PO 文件中的已翻译字符串。在进行合并前会把所有文件都自动转换为 UTF-8 编码。

当 INPUT 为一个 XML 文件时，输出的 OUTPUT 文件也将是 XML，PO 文件中翻译的字符串将作为 `xml:lang` 属性插入到原 XML 中一并写入 OUTPUT 文件。

4.2.3 xml2po：双向转换 Gtk XML 和 PO

xml2po 是 gnome-doc-utils 中的一个工具，主要用途是在 XML 和 PO 文件之间进行转换和合并。

从 XML 文件生成 POT 文件：

```
xml2po -e -o book.pot book.xml
```

将已翻译的 zh_CN.po 合并回原来的 XML 文件中：

```
/usr/bin/xml2po -e -p zh_CN.po -o book.zh_CN.xml book.xml
```

后记

本文是以 KDE 中国站点上刊载的 I18N/L10N 工作流程为蓝本所修订的工作指南，过程中参考了 i18n-zh 以及 GNOME Live 上的一些文章，还包含了 iKDE 上 csslayer 的如何翻译 KDE 中的部分内容，后来又补充了在繁体中文分支中做出的一些改动。欢迎对此指南的内容进行讨论，请将你的意见和建议发送至 i18n-zh@googlegroups.com，或者 https://github.com/AOSC-Dev/translations/issues 也可以。

除了之前提到的《中文文案排版指北》，ReadTheDocs 上的通用《LocalizationGuide》（英语，适用多语言，推荐）和微软《简体中文格式指南》（英语）也很值得参考。W3C 另有一份《中文排版需求》，在符号使用等信息方面较为全面。

要获得更多的参考信息，可以看看这些地方：

- https://people.ubuntu.com/~happyaron/l10n/
- https://translatehouse.org
- http://ftp.asia.edu.tw/cpatch/g/glossary/
- https://repo.anthonos.org/misc/l10n/ & https://github.com/AOSC-Dev/translations/

本文档使用了 FandolSong、Inziu Iosevka SC 和思源黑体 CN，可能是现在能找到的最棒的一组自由版权的中文电脑字体了吧。当然，Liberation Serif 也是很重要的。格式嘛是 Hybrid PDF，想接手的话用 LibreOffice 或者 OpenOffice 打开编辑就可以了。如果对 ODT 版本感兴趣的话，这里也有。

文档现状

本文档并非 zh_CN 主流翻译小组之标准文档，在与已存在之项目协作时仍请慎重使用，避免争端。

最早的初始 Google Docs 发行版本，有以下几个直接分支：

1. AOSC 你所看到的版本。
2. zh_Hant 繁体中文的版本。
3. UbuntuCN Ubuntu 中文化团队——其实总归都是同一伙人啦——使用的“正统”版本。

目前分支（AOSC）的作者正在努力将更改并入以上其他版本。