.Deb

銀河系唯一のDebian専門誌

2022 年 5 月 21 日

gettext特集

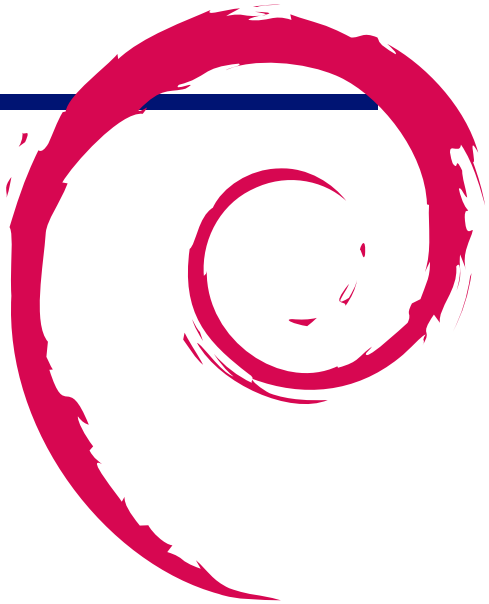
デビアン勉強会

目次

1 最近の Debian 関連のミーティング報告 2
1.1 2022 年 4 月度 東京エリア・関西合同 Debian 勉強会 . . . 2

2 事前課題 3
2.1 dictoss 3
2.2 yy_y_ja_jp 3
2.3 yosukesan (yosuke_san) . . 3
2.4 koedoyoshida 3
2.5 NOKUBI Takatsugu (knok) 3
2.6 ysaito 3
2.7 Yamada Yohei (yoheiy) . . 3
2.8 Kazuhiro NISHIYAMA (znz) 3
2.9 YukiharuYABUKI 4
2.10 Hiroyuki Yamamoto (yama1066) 4
2.11 レドレ (redred) 4
2.12 gdevmjc 4
2.13 Takayoshi Shiigi (tcgi) . . 4
2.14 yukiy99999999999999999999999 4
2.15 jsynth21 4

3 アプリにおける多言語対応の仕組み 5
3.1 はじめに 5
3.2 Debian における I18N と L10N について 5
3.3 locale 5
3.4 gettext と メッセージカタログ 6
3.5 C 言語のアプリケーションを例にした翻訳処理の実装例 . 6
3.6 終わりに 10
3.7 参考文献 10



1 最近の Debian 関連のミーティング報告

杉本 典充

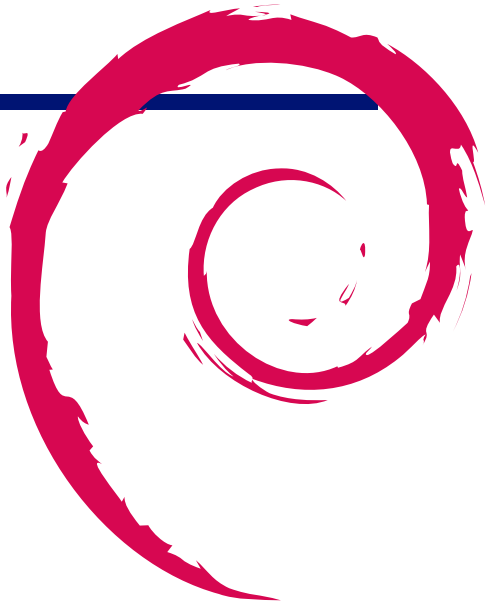
1.1 2022 年 4 月度 東京エリア・関西合同 Debian 勉強会

2022 年 4 月 16 日 (土) に東京エリア Debian 勉強会と関西 Debian 勉強会の合同でオンラインによる Debian 勉強会を開催しました。参加者は 8 名でした。

セミナー発表は dictoss さんの「DDTP 及び DDTSS の紹介」を行い、BoF「Debian のおすすめポイント」では参加者同士で意見を出し合いました。BoF でディスカッションした議事録は以下 URL で公開しています。

- `https://tokyodebian-team.pages.debian.net/2022-04_tokyodebian_bof.txt`

勉強会の終了後、参加者同士で Debian や OSS に関する話の情報交換を行いました。



2 事前課題

杉本 典充

今回の事前課題は以下です。

1. 使っているメールクライアントソフトを教えてください
2. OSS の活動で利用しているメールサービスを教えてください
3. メールを送受信するために利用している認証方式を教えてください
4. メールを送受信するためのパスワードやトークンをどこに保存していますか

2.1 dictoss

1. Web ブラウザ （Web メール）, スマートフォンのメールアプリ, Sylpheed / Claws Mail
2. outlook / hotmail, 自分でメールサーバを立てて利用
3. アカウント名とパスワード
4. メールクライアントソフトのアカウント設定に保存

2.2 yy_y_ja_jp

1. Web ブラウザ （Web メール）, Thunderbird, その他
2. gmail, 自分でメールサーバを立てて利用, その他
3. アカウント名とパスワード, OAuth
4. 毎回入力している (web メールを利用), メールクライアントソフトのアカウント設定に保存, その他

2.3 yosukesan (yosuke_san)

1. Web ブラウザ （Web メール）
2. gmail
3. アカウント名とパスワード
4. パスワードマネージャーソフトに保存

2.4 koedoyoshida

1. Becky! Internet Mail
2. gmail
3. アカウント名とパスワード, OAuth
4. メールクライアントソフトのアカウント設定に保存

2.5 NOKUBI Takatsugu (knok)

1. Web ブラウザ （Web メール）, スマートフォンのメールアプリ, Sylpheed / Claws Mail, Mew / Wanderlust
2. gmail, 自分でメールサーバを立てて利用
3. アカウント名とパスワード
4. 毎回入力している (メールクライアントソフトを利用), その他

2.6 ysaito

1. Web ブラウザ （Web メール）, スマートフォンのメールアプリ, Thunderbird
2. gmail
3. アカウント名とパスワード
4. 毎回入力している (web メールを利用), メールクライアントソフトのアカウント設定に保存

2.7 Yamada Yohei (yoheiy)

1. スマートフォンのメールアプリ, Microsoft Outlook
2. gmail
3. その他
4. メールクライアントソフトのアカウント設定に保存, その他

2.8 Kazuhiro NISHIYAMA (znz)

1. Web ブラウザ （Web メール）, スマートフォンのメールアプリ, Mew / Wanderlust

2. gmail, ISP のメールサービス, 自分でメールサーバを立てて利用
3. アカウント名とパスワード, OAuth, 認証なし
4. メールクライアントソフトのアカウント設定に保存, パスワードマネージャーソフトに保存, ハードウェアトークンに保存, その他

2.9 YukiharuYABUKI

1. Web ブラウザ （Web メール）, スマートフォンのメールアプリ, Sylpheed / Claws Mail, Mutt
2. gmail, ISP のメールサービス, 自分でメールサーバを立てて利用
3. アカウント名とパスワード
4. メールクライアントソフトのアカウント設定に保存, その他

2.10 Hiroyuki Yamamoto (yama1066)

1. Web ブラウザ （Web メール）, Thunderbird
2. gmail
3. アカウント名とパスワード
4. 毎回入力している (web メールを利用), メールクライアントソフトのアカウント設定に保存

2.11 レドレ (redred)

1. Web ブラウザ （Web メール）
2. gmail, yahoo
3. アカウント名とパスワード
4. 毎回入力している (web メールを利用)

2.12 gdevmjc

1. Web ブラウザ （Web メール）, スマートフォンのメールアプリ, Mew / Wanderlust
2. gmail
3. アカウント名とパスワード, その他
4. 毎回入力している (メールクライアントソフトを利用), パスワードマネージャーソフトに保存

2.13 Takayoshi Shiigi (tcgi)

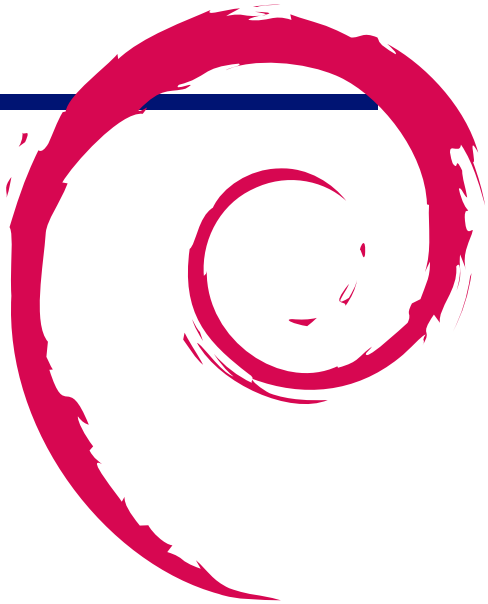
1. Web ブラウザ （Web メール）, Sylpheed / Claws Mail, Mew / Wanderlust
2. gmail
3. アカウント名とパスワード
4. メールクライアントソフトのアカウント設定に保存

2.14 yukiy99999999999999999999999999

1. Web ブラウザ （Web メール）
2. gmail
3. アカウント名とパスワード
4. 毎回入力している (メールクライアントソフトを利用)

2.15 jsynth21

1. Thunderbird
2. gmail
3. アカウント名とパスワード
4. メールクライアントソフトのアカウント設定に保存



3 アプリにおける多言語対応の仕組み

杉本 典充

3.1 はじめに

アプリケーションを国際化するには出力する文字列を各言語に翻訳して出力する必要があります。今回はプログラムの実行時に翻訳した文字列に置換する gettext とメッセージカタログについて調べてみました。

3.2 Debian における I18N と L10N について

Debian リファレンスに「第 8 章 I18N と L10N」の記事があります[*1]。この記事では I18N と L10N は以下のように説明しています。

- 国際化 (I18N[*2]): ソフトが複数のロケール (地域) を扱えるようにします。
- 地域化 (L10N[*3]): 特定のロケール (地域) を扱えるようにします。

また、Debian デベロッパー レファレンスに「8. 国際化と翻訳」の記事があります[*4]。この記事では「あなたが英語圏のネイティブスピーカーで他の言語を話さないとしても、国際化の問題について注意を払うことはメンテナとしてのあなたの責務です」と記述があります。そのため、Debian Developer を目指すにはプログラムの国際化についてどのような処理を行っているか知っておく必要があります。

3.3 locale

自分が使っているコンピュータを地域化 (L10N) した状態で動作させるには使う言語を設定する必要があります。その設定を locale (ロケール) といい、Debian や 多くの Linux システム では C ロケール (POSIX ロケールともいわれる) を用いています。Debian における locale 設定は Debian インストーラや ”dpkg-reconfigure locales” コマンドで設定でき、locale を設定すると、LANG 環境変数に locale 名が設定されます。

なお、C ロケールは次の記法で地域を特定する仕様になっています。

- (言語コード 2 文字)_(国コード 2 文字).(文字コード)

言語コード 2 文字は ISO-639、国コード 2 文字は ISO-3166 を用いることになっています。文字コードは Debian では国際化するために内部処理で扱う文字コードは UTF-8 とすることが一般的になっているため、”UTF-8” を指定することが多いと思います。

[*1] `https://www.debian.org/doc/manuals/debian-reference/ch08.ja.html`
[*2] internationalization のこと。文字数が長いため簡略した記述として使われる。
[*3] localization のこと。文字数が長いため簡略した記述として使わる。
[*4] `https://www.debian.org/doc/manuals/developers-reference/l10n.ja.html`

このルールに従い日本、アメリカ、イギリス、カナダの locale 名は以下のようになります。

表 1　locale の表記補法

国名	言語コード	国コード	locale 名
日本	ja	JP	ja_JP.UTF-8
アメリカ	en	US	en_US.UTF-8
イギリス	en	GB	en_GB.UTF-8
カナダ	en	CA	en_CA.UTF-8
カナダ	fr	CA	fr_CA.UTF-8

そのほか、locale 名には ”C” というデフォルトのロケール設定[*5]も存在します。

3.4　gettext と メッセージカタログ

Debian では国際化及び地域化の処理を行う gettext パッケージ、po4a パッケージ、poedit パッケージなどを提供しています。

gettext パッケージはソフトウェアの中に埋め込んだ英語で記述した文字列を地域化した文字列に動的に置換して出力するライブラリを含んでいます。この gettext の仕組みを使うことで、プログラムの実行時に locale の値に従って地域化した翻訳文字列を表示することができます。

なお、地域化した文字列を保存するファイルを「メッセージカタログ」と呼びます。メッセージカタログは 3 つの形式があり、それぞれ異なる拡張子をもちます。

- pot ファイル
  - xgettext コマンドでソースコードから翻訳が必要な文字列を抽出した po ファイルのテンプレートとなるテキストファイル
- po ファイル
  - msginit コマンドで pot ファイルから生成する各言語の翻訳結果を保存するテキストファイル
- mo ファイル
  - msgfmt コマンドで po ファイルをコンパイルして生成するバイナリファイルで、プログラムが参照する

3.5　C 言語のアプリケーションを例にした翻訳処理の実装例

3.5.1　ディレクトリとファイルの配置

まずは C 言語のソースコード main.c と手書きした Makefile を作成します。

```
$ cd myhello
$ find .
.
./src
./src/main.c
./Makefile
```

例として用意した src/main.c と Makefile は以下です。

[*5] LANG=C と書いた方が馴染みがあるかもしれません。

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

#define _(String) gettext(String)

#define PACKAGE_NAME "myhello"
#define LOCALE_DIR   "locale"

int main(int argc, char *argv[])
{
    char *modir = NULL;

    setlocale(LC_ALL, "");
    textdomain(PACKAGE_NAME);
    modir = bindtextdomain(PACKAGE_NAME, LOCALE_DIR);

    printf("modir: %s\n", modir);
    printf("%s\n", _("Hello world."));

    return 0;
}
```

```
$ cat Makefile

#
# Makefile for myhello
#
PROG = myhello
CFLAGS =
LDFLAGS =

SRCS = src/main.c
OBJS = $(SRCS:.c=.o)

PO_SRCS = po/ja.po
PO_OBJS = $(PO_SRCS:.po=.mo)


.PHONY: clean pot poinitja pomergeja poinstallja

all: $(PROG)

$(PROG): $(OBJS) $(PO_OBJS)
        $(CC) $(LDFLAGS) -o $@ $(LIBS) $(OBJS)

clean:
        $(RM) $(OBJS) $(PO_OBJS) *~

pot:
        xgettext -k"_" -o $(PROG).pot $(SRCS)

poinitja:
        msginit --locale ja_JP.UTF-8 --input $myhello.pot --output-file po/ja.po

pomergeja:
         msgmerge -U po/ja.po myhello.pot

poinstallja:
        install -d locale/ja/LC_MESSAGES
        install $(PO_OBJS) locale/ja/LC_MESSAGES/${PROG}.mo

.SUFFIXES: .c .o .po .mo

.c.o:
        $(CC) $(DEBUG) $(CFLAGS) -c -o $@ $<

.po.mo:
        msgfmt -o $@ $<
```

3.5.2 locale と gettext の処理

locale 処理の関数を呼び出すために locale.h、gettext() 関数を呼び出すために libintl.h を include します。
呼び出す locale 処理の関数には以下の意味があります。

- setlocale(LC_ALL, "");
  - プログラムの locale を設定する
  - 第 1 引数の LC_ALL は、日付や金額などのロケールのカテゴリすべての値を変更する
  - 第 2 引数は指定する locale で空文字を設定した場合はプログラム実行時の LANG 環境変数を設定する
- textdomain(PACKAGE_NAME);
  - プログラムのドメイン名を設定する
  - 第 1 引数はドメイン名で、通常はプログラム名と同じ文字列を指定する (今回の場合は "myhello")

- modir = bindtextdomain(PACKAGE_NAME, LOCALE_DIR);
  - 第 1 引数はドメイン名で、通常はプログラム名と同じ文字列を指定する (今回の場合は ”myhello”)
  - 第 2 引数は参照するコンパイルしたメッセージカタログ (mo ファイル) を検索する起点のディレクトリを設定する[*6]
  - 今回の例では、ドメイン名は ”myhello” のため myhello.mo を 相対パスの ”locale” ディレクトリから探す、という意味になる

翻訳した文字列に置換する処理を行う gettext の仕組みでは C 言語のマクロの _(String) の String 部分が置換対象になります。

```
    printf("%s\n", _("Hello world."));
```

メッセージカタログを準備していない状態でコンパイルしてプログラムを実行すると、”Hello world.” と英語の文字列が出力されます。

```
$ make
cc -c -o src/main.o src/main.c
cc -o myhello  src/main.o

$ ./myhello
podir: locale
Hello world.
```

3.5.3 メッセージカタログの準備

まずは xgettext コマンドを実行して pot ファイルを生成します。pot ファイルは各言語のメッセージカタログのテンプレートとなるファイルのため、Language や charset などが未定義になっています。

```
$ xgettext -k"_" -o myhello.pot src/main.c
$ file myhello.pot
myhello.pot: GNU gettext message catalogue, ASCII text

$ cat myhello.pot
# SOME DESCRIPTIVE TITLE.
# Copyright (C) YEAR THE PACKAGE'S COPYRIGHT HOLDER
# This file is distributed under the same license as the PACKAGE package.
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
#
#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2022-05-19 21:32+0900\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"Language: \n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: 8bit\n"

#: src/main.c:19
msgid "Hello world."
msgstr ""
```

次に msginit コマンドを実行して各言語の po ファイルを生成します。今回は日本語用の ja.po を生成してみます。各言語の po ファイルは po ディレクトリにまとめて保存することが多いため、po ディレクトリを作成しています。

[*6] debian パッケージを作成する場合は /usr/share/locale 配下のメッセージカタログをインストール仕様になっています。

```
$ mkdir po
$ msginit --locale ja_JP.UTF-8 --input myhello.pot --output-file po/ja.po
ユーザが翻訳に関するフィードバックをあなたに送ることができるように，
新しいメッセージカタログにはあなたの email アドレスを含めてください．
またこれは，予期せぬ技術的な問題が発生した場合に管理者があなたに連絡が取れる
ようにするという目的もあります．

Is the following your email address?
  dictoss@localhost
Please confirm by pressing Return, or enter your email address.
dictoss@live.jp
https://translationproject.org/team/index.html を検索中... 完了．
Please visit your translation team's homepage at
  https://translationproject.org/team/ja.html
  https://translationproject.org/team/index.html
  https://translationproject.org/html/translators.html
  https://translationproject.org/html/welcome.html
and consider joining your translation team's mailing list
  <translation-team-ja@lists.sourceforge.net>

po/ja.po を生成．
```

生成した po/ja.po ファイルは以下となります。

```
$ file po/ja.po
po/ja.po: GNU gettext message catalogue, UTF-8 Unicode text

$ cat po/ja.po

# Japanese translations for PACKAGE package
# PACKAGE パッケージに対する英訳．
# Copyright (C) 2022 THE PACKAGE'S COPYRIGHT HOLDER
# This file is distributed under the same license as the PACKAGE package.
# Norimitsu SUGIMOTO <dictoss@live.jp>, 2022.
#
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2022-05-20 23:37+0900\n"
"PO-Revision-Date: 2022-05-21 10:53+0900\n"
"Last-Translator: Norimitsu SUGIMOTO <dictoss@live.jp>\n"
"Language-Team: Japanese <translation-team-ja@lists.sourceforge.net>\n"
"Language: ja\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"
"Plural-Forms: nplurals=1; plural=0;\n"

#: src/main.c:19
msgid "Hello world."
msgstr ""
```

po/ja.po ファイルを編集し、原文の英語を日本語に翻訳します[*7]。

```
#: src/main.c:19
msgid "Hello world."
msgstr "こんにちは世界。"
```

次に msgfmt コマンドを実行して po ファイルから mo ファイルを生成します。

```
$ msgfmt -o po/ja.mo po/ja.po
$ file po/ja.mo
po/ja.mo: GNU message catalog (little endian), revision 0.0, 2 messages,
Project-Id-Version: PACKAGE VERSION '\343\201\223\343\202\223\343\201\253\343\201
\241\343\201\257\344\270\226\347\225\214\343\200\202'
```

この状態で myhello コマンドを実行してみても、プログラムの出力文字列は翻訳された文字列になりません。

```
$ echo $LANG
LANG=ja_JP.UTF-8

$ ./myhello
podir: locale
Hello world.
```

プログラムが読み込む mo ファイルのディレクトリとファイル名は bindtextdomain() の引数で決まります。今回は bindtextdomain("myhello", "locale") の引数で呼び出しているため mo ファイルはプログラム実行時のカレント

[*7] vi などでファイルを編集してもよいですが、poedit パッケージに含まれる poeditor というグラフィカルな翻訳エディタを使うと便利です。

ディレクトリからの相対パス ”locale/ja/LC_MESSAGES/myhello.mo” を読み込みます[*8]。

なお、debian でパッケージを作成してプログラムを提供する場合は mo ファイルを 以下のディレクトリ及びファイル名で配置する必要があります。

- /usr/share/locale/(言語コード 2 文字)/LC_MESSAGES/(プログラム名).mo

```
$ mkdir -p locale/ja/LC_MESSAGE
$ cp -p po/ja.mo locale/ja/LC_MESSAGES/myhello.mo
```

この状態で myhello コマンドを実行してみると、プログラムの出力文字列は翻訳した日本語を出力できました。

```
$ echo $LANG
LANG=ja_JP.UTF-8

$ ./myhello
podir: locale
こんにちは世界。
```

3.5.4 ソースコードを変更した場合のメッセージカタログの更新

ソースコードを変更して翻訳文字列が増える場合や減る場合、行番号がずれる場合があります。その場合は xgettext コマンドで pot ファイルを再生成し、msgmerge コマンドで既存の po ファイルと 再生成した pot ファイルをマージします。

msgmerge コマンドでマージしたときにマージに失敗した翻訳文字列は、その翻訳文字列の直前に ”fuzzy” という目印がつきます。

```
$ vi src/main.c
     printf("%s\n", gettext("Hello world."));
+    printf("%s\n", gettext("Hello earth."));

$ xgettext -k"_" -o myhello.pot src/main.c
$ msgmerge -U po/ja.po myhello.pot
... 完了.

$ diff -u po/ja.po~ po/ja.po
--- po/ja.po~   2022-05-20 23:21:29.263856024 +0900
+++ po/ja.po    2022-05-20 23:37:58.014853417 +0900
@@ -8,7 +8,7 @@
 msgstr ""
 "Project-Id-Version: myhello 0.0.1\n"
 "Report-Msgid-Bugs-To: \n"
-"POT-Creation-Date: 2022-05-20 21:57+0900\n"
+"POT-Creation-Date: 2022-05-20 23:37+0900\n"
 "PO-Revision-Date: 2022-05-19 21:43+0900\n"
 "Last-Translator: Norimitsu SUGIMOTO <dictoss@live.jp>\n"
 "Language-Team: Japanese <translation-team-ja@lists.sourceforge.net>\n"
@@ -21,3 +21,8 @@
 #: src/main.c:19
 msgid "Hello world."
 msgstr "こんにちは世界。"
+
+#: src/main.c:20
+#, fuzzy
+msgid "Hello earth."
+msgstr "こんにちは世界。"
```

3.6 終わりに

C 言語のプログラムで地域化の翻訳処理を行うにあたり、メッセージカタログの取り扱い方法を説明しました。gettext を使いこなしつつメッセージカタログの翻訳を行ってプログラムの国際化を目指しましょう。

3.7 参考文献

- Debian Wiki - Locale `https://wiki.debian.org/Locale`
- Weblate - GNU gettext を使用したソフトウェアの翻訳
  `https://docs.weblate.org/ja/latest/devel/gettext.html`

[*8] ja の部分は言語名になります。例えば LANG 環境変数がフランス語の場合は fr になります。

Debian 勉強会資料

2022 年 5 月 21 日　　初版第 1 刷発行

東京エリア Debian 勉強会 （編集・印刷・発行）