



Grand Unified Debian



銀河系唯一のDebian専門誌

東京エリア/関西Debian勉強会



あんどきゅめんとでっど でびあん 2022年冬号 2022年12月31日 初版発行

Android 勉強会

目次

1	Introduction	2
2	BoF Debian のおすすめポイント	3
3	Debian で Android アプリ開発と Kotlin	6
4	git-buildpackage を使ってみる	8
5	DDTP 及び DDTSS の紹介	16
6	アプリにおける多言語対応の仕組み	18
7	Gmail とパスワード	24

1 Introduction

DebianJP



1.1 東京エリア Debian 勉強会

Debian 勉強会へようこそ。これから Debian の世界にあしを踏み入れるという方も、すでにどっぷりとつかっているという方も、月に一回 Debian について語りませんか？

Debian 勉強会の目的は下記です。

- Debian Developer (開発者) の育成。
- 日本語での“開発に関する情報”を整理してまとめ、アップデートする。
- 場の提供。
 - 普段ばらばらな場所にいる人々が face-to-face で出会える場を提供する。
 - Debian のためになることを語る場を提供する。
 - Debian について語る場を提供する。

Debian の勉強会ということで究極的には参加者全員が Debian Package をがりがりとするスーパーハッカーになった姿を妄想しています。情報の共有・活用を通して Debian の今後の能動的な展開への土台として、“場”としての空間を提供するのが目的です。

1.2 関西 Debian 勉強会

関西 Debian 勉強会は Debian GNU/Linux のさまざまなトピック (新しいパッケージ、Debian 特有の機能の仕組、Debian 界隈で起こった出来事、などなど) について話し合う会です。

目的として次の三つを考えています。

- ML や掲示板ではなく、直接顔を合わせる事での情報交換の促進
- 定期的集まれる場所
- 資料の作成

それでは、楽しい一時をお楽しみ下さい。

2 BoF Debian のおすすめポイント

2022 年 4 月の参加者



2.1 参加者

- dictoss
- knob
- yy-y-ja.jp
- sirtetris
- redred
- Hiroyuki Yamamoto (yama1066)
- tcgi
- ipv6waterstar

2.1.1 利用者として Debian を使う良さ

- フリー (自由) などころ
- 移植版が多く、いろいろな CPU で動作する
- 今でも 32bit を明確にサポートしている
- PC、サーバ、組み込み機器で動くため OS を統一でき利用者には便利なこと
- メモリ量が少ない、ディスク容量が少ないハードウェアにもインストールできる (Debian 10 bullseye ではメモリは最小 256MB でインストール可能)
- 使いやすい、かつしっかりしたパッケージ管理
- パッケージ量が多く、使いたいアプリケーションのパッケージはだいたいある
- Debian Science チームが結構多くのフリーな科学関連パッケージを入れてくれている
- stable-backports があるため、安定版でも最近の新しいアプリケーションやカーネルが提供されている
- アップグレードが保証されていること (ここ具体的に知りたいです)
 - アップグレードができるようなパッケージを作成するようポリシーで定められている
 - ただし複数のメジャーバージョンを跨いだアップグレードは保証しない (順番に 1 つずつメジャーバージョンをアップグレードすれば最新バージョンまで上げることができる)
- 派生が多いということは、それだけ多くの人の目が通っていて、安全な OS なのだろうと思える安心感
- Debian に基づくディストリビューションが多いから、新しいパッケージマネージャーなどを学ばず割と簡単に Debian に移れる (例えば初心者向けのイメージがある Ubuntu から)
- 日常のアップデート頻度が Ubuntu より控えめな気楽さ
- 簡単にサーバを構築できる

- 因みに、我が家には 10 台ぐらいのサーバが動いている
- 異なるディストリビューションが用意されている (stable/testing/unstable)
- ポリシーがしっかりしているところ
- セキュリティ情報がしっかり提供されている
- セキュリティの脆弱性情報が公開された後すぐに更新パッケージが配布されることが多い

2.1.2 開発者として Debian を使う良さ

- 中身が公開されているので自分の手元で直せる
- フリー (自由) のため再利用や改変などしてよいことが保証されている
- 個々のパッケージのライセンスを確かめやすい (フォーマット化されたライセンス情報)
 - <https://sources.debian.org/> を参照すればローカルにファイルがなくてもよい
- 「Debian 系」と言われるように Debian 系の派生 OS の源流のため色々な情報が集まる
- buildd (<https://build.debian.org/>) で、各アーキテクチャのビルドログなどを参照できる
- 開発者の身元確認が厳格で、OpenPGP による Web of Trust が構築されている

2.1.3 Debian が他のディストリビューションから知見を取り入れるとよいこと

- 敷居？
 - 他のディストリビューションのように気楽に Debian を使ったり開発したりしてほしいが何が障害なのだろうか？
 - 気軽に Debian を使ってみる機会があるとよいのでは？
 - * 気軽に触れる Web サービス等 (ビルドサービス OBS とか) があるといい？
 - * ブラウザで触れる Debian 環境 WebVM <https://webvm.io/>
- Secure Boot 関連
 - Debian 10 buster (2019-07-06 リリース) から Secure Boot に対応している
 - * ただし DKMS を使って自分でビルドしたカーネルモジュールは Secure Boot すると使えない仕様になっている
 - 他のディストリビューションはどうなのでしょう？
- リリース時期、サポート期限のコミット発表時期がいつと決まっていない、リリースと同時にサポート期限が公開されない
 - リリース日がいつになるかがかなり前から事前にわからないのは確かにそのとおり
 - * Debian のリリースは Time-Based Release ではないことが理由。ただし、およそ 2 年ごとにリリースするように運営している。
 - * 「Debian は時間ベースのリリースフリーズを採用します」 <https://www.debian.org/News/2009/20090729>
 - stable のサポート期限は 3 年と決まっている
 - * <https://www.debian.org/releases/index.ja.html>
 - stable には LTS を提供している (intel 系/arm 系に限定して提供する実情あり)
 - * <https://wiki.debian.org/LTS/Extended>
 - * <https://deb.freexian.com/extended-lts/>
 - 日本語のドキュメントが他のディストリビューションより少ない気がする
 - * Debian wiki はだいたい英語な気がする
 - * Arch wiki はかなり日本語訳が豊富
 - ・ どのような仕組みでやっているのでしょうか。単に人が多いから？敷居？
 - 時々変わったものを採用したりする

- * wayland
 - Debian 10 buster (2019-07-06 リリース) で wayland は GNOME を使うと使えるようになった
 - <https://www.debian.org/releases/buster/amd64/release-notes/ch-whats-new.ja.html#wayland-by-default-on-gnome>
 - 上流の GNOME が wayland をデフォルトに変更したため、Debian もそのデフォルトを取り込んだ経緯がある
 - ただ、他のウィンドウマネージャは継続して X11 を利用するようになっている
- * mariadb
 - Debian 9 stretch (2017-06-17 リリース) で mysql から mariadb を採用するように変更した
 - <https://www.debian.org/releases/stretch/amd64/release-notes/ch-whats-new.ja.html#mariadb-replaces-mysql>
- * Exim4
 - Debian では MTA のデフォルトに Exim4 を採用している
- ウィンドウマネージャの選択肢が多いのが問題？
 - パッケージの多様性がどれを選べばよいかわからなくしている？
 - 直接は関係ないが、いまだに Compiz が残ってる

3 Debian で Android アプリ開発と Kotlin

杉本 典充



3.1 はじめに

最近 Android アプリを開発しており、Debian で Android アプリを開発するにはどのような準備をすればよいのか、最近の Android アプリの開発で使われるプログラム言語「Kotlin」の Debian における対応状況について調べてみました。

3.2 Debian で Android アプリを開発する

3.2.1 Android 情報のドキュメントの場所

Android アプリの開発者向けのドキュメントは <https://developer.android.com/?hl=ja> にあります。

この Web ページでは Android アプリの開発に利用する Android Studio のダウンロードとインストール方法、最近の Android 情報や開発者ガイド、ガイドライン、チュートリアルなどの情報を公開しています。

3.2.2 Android Studio のインストール

Linux 環境に Android Studio をインストールする手順は、開発者向けドキュメントの <https://developer.android.com/studio/install?hl=ja#linux> に記載があり、Debian Wiki^{*1} にも情報があります。インストール手順には tarball をダウンロードして任意のディレクトリに展開する方法と、snap 環境にインストールする方法があります。

ここでは、<https://developer.android.com/studio#downloads> から利用規約を読み承諾してダウンロードし^{*2}、展開して Android Studio を起動するコマンドを示します。

```
$ tar xf android-studio-2020.3.1.26-linux.tar.gz
$ cd android-studio/bin
$ bash studio.sh
```

studio.sh を起動すると、Android Studio の画面が表示され、実行に必要な追加のソフトウェアを大量にダウンロードします。ダウンロードとインストールが完了すると Android Studio を起動できます。

なお、Android Studio は IntelliJ IDEA ベースで作られています。そのため Android Studio を実行するには Java 環境が必要であり tarball の中に OpenJDK 11 が含まれています (Debian が提供する openjdk-11-jre パッケージをインストールせずに実行できます)。

^{*1} <https://wiki.debian.org/AndroidStudio>

^{*2} ダウンロードする tarball は 935 MiB ありますので、定額で高速にインターネット接続できる環境でセットアップすることをおすすめします。

3.2.3 Android エミュレータを実行するために KVM を設定する

Android アプリをデバッグするために Android のエミュレータ (=仮想マシン) が提供されています。Android のエミュレータのイメージファイルは qemu で動作する仮想マシンになっているため、PC 上で KVM を使えるように以下のパッケージをインストールします。

```
# apt-get update
# apt-get install qemu-kvm
```

3.2.4 Android アプリを実機でデバッグ実行できるように設定する

Android Studio ではビルドした Android アプリをデバッグ実行することができます。デバッグ実行すると、設定したエミュレータ上または Android スマートフォンの実機上でアプリを動かすことができます。Android アプリを Android スマートフォンの実機上で実行する手順は、<https://developer.android.com/studio/run/device?hl=ja> に説明があります。

Debian では PC と Android スマートフォンの実機を USB 接続したことを検知するためのパッケージを提供しており、android-sdk-platform-tools-common パッケージをインストールすると udev の設定ファイルである "/lib/udev/rules.d/51-android.rules" をインストールします。パッケージのインストール後は、udev の設定ファイルを再読み込みします。なお、USB 接続する Android スマートフォンが Pixel シリーズの場合は、Debian 11 bullseye が提供する linux kernel にドライバが含まれているため、別途ドライバのインストールは不要です。

```
# apt-get install android-sdk-platform-tools-common
# systemctl reload udev
```

次に Android スマートフォンを設定します。Android スマートフォンの設定画面を開いて開発者向けオプションを有効にし^{*3}、開発者向けオプションの設定画面にある「USB デバッグ」を有効にします。この状態の Android スマートフォンを Android Studio を実行している PC に USB 接続すると、Android スマートフォンの画面にフィンガープリントを確認する画面が出てきますので「はい」を選択します。少し待つと Android Studio の上部にある Android アプリの実行対象デバイスを表示するメニューに Android スマートフォン実機の型番が表示されます。ここまで準備ができると Android Studio でアプリをデバッグ実行すれば 実機で Android アプリを動かすことができます。

3.3 Debian での Kotlin

3.3.1 ドキュメントの情報

Kotlin の upstream の Web サイトは <https://kotlinlang.org/> にあり、ソースコードは Apache License 2.0 の下でソースコードを公開しています^{*4}。2021 年 12 月 18 日現在では、バージョン 1.6.10 を提供しています。

Debian における Kotlin の情報は、<https://wiki.debian.org/Kotlin> に記載があります。Debian パッケージでは unstable のみですが kotlin パッケージを提供しており (kotlin-1.3.31)、<https://salsa.debian.org/java-team/kotlin> で開発作業を進めています。upstream のバージョンと比べると Debian パッケージのものはだいぶ遅れている状況ですが、セルフビルドを目標に作業を進めています。

3.4 おわりに

Debian で Android アプリを開発するために Android Studio を設定してみました。Kotlin を Debian で使うにはまだ発展途上の状況ですが、ぜひ新しいバージョンが利用できるようになってほしいです。

^{*3} 「ビルド番号」の項目を 7 回連続でタップすると開発者向けオプションが有効になります。 <https://developer.android.com/studio/debug/dev-options?hl=ja>

^{*4} <https://github.com/JetBrains/kotlin>

4 git-buildpackage を使ってみる

杉本典充



4.1 はじめに

debian パッケージを作成し管理する方法はいろいろあると思います。

今回は git でパッケージを管理し、ビルドする仕組みを提供する git-buildpackage (gbp) について調べてみました。

4.2 git-buildpackage とは

debian では「git-buildpackage」パッケージを提供しています。sid では git-buildpackage-0.9.25 を提供しており、gbp コマンド*⁵、git-pbuilder コマンドが含まれています。

gbp コマンドには debian パッケージを開発する上で様々な操作ができる便利な以下のサブコマンドがあります*⁶。

- gbp buildpackage
- gbp import-orig
- gbp export-orig
- gbp import-dsc,dscs
- gbp import-ref
- gbp dch
- gbp pq
- gbp pull,clone,push
- gbp create-remote-repo
- gbp tag
- gbp config
- gbp pristine-tar
- gbp setup-gitattributes

なお、git-buildpackage というパッケージ名になっていますが、他のバージョン管理システムに対応した cvs-buildpackage、svn-buildpackage、mercurial-buildpackage パッケージも存在します。

*⁵ 中身は python3 で書かれています。

*⁶ `$ gbp --list-cmds`

4.3 git-buildpackage を使って tarball を新規にパッケージ化する

ここでは例として、GNU hello^{*7} の tarball を debian パッケージ化^{*8}する作業を gbp コマンドを使って行ってみます。

なお、今回の作業環境は Debian sid とします。

4.3.1 必要な debian パッケージのインストール

まずは git-buildpackage パッケージを apt-get コマンドでインストールします。そして、debian パッケージの作成に必要な build-essential、大変便利な debhelper をインストールします。

```
$ sudo apt-get update
$ sudo apt-get install git-buildpackage
$ sudo apt-get install build-essential debhelper dh-make
```

今回 Debian パッケージを作成する GNU hello のビルドに使うツール、およびビルドに必要な依存パッケージをインストールしておきます。

```
$ sudo apt-get install wget
$ sudo apt-get install gnulib help2man
```

4.3.2 git リポジトリの作成と tarball のインポート

まずは GNU hello の tarball をダウンロードします。

```
$ wget https://ftp.gnu.org/gnu/hello/hello-2.10.tar.gz
```

git のローカルブランチのディレクトリ名は tarball のファイル名と同じ "hello" にすることとし、git リポジトリ "hello" をローカルに作成します。最近の git の使い方としてデフォルトを main ブランチにするようになってきていますので、ブランチ名を指定しています^{*9}。

```
$ git init hello --initial-branch=main
$ cd hello
```

tarball を指定して gbp import-orig コマンドを実行すると、tarball を git に取り込んでコミットします。デフォルトでは master ブランチを作成しますが、<https://salsa.debian.org/> の git デフォルトブランチは main のため `-debian-branch=main` を指定しています。

```
$ gbp import-orig --debian-branch=main --pristine-tar ../hello-2.10.tar.gz
What will be the source package name? [hello]
What is the upstream version? [2.10]
gbp:info: Importing '../hello-2.10.tar.gz' to branch 'upstream'...
gbp:info: Source package is hello
gbp:info: Upstream version is 2.10
gbp:info: Successfully imported version 2.10 of ../hello_2.10.orig.tar.gz
```

gbp import-orig で tarball を取り込むと以下のブランチとタグが作成されます。

```
$ git branch
* main
  pristine-tar
  upstream
```

```
$ git tag
upstream/2.10
```

^{*7} <https://www.gnu.org/software/hello/>

^{*8} <https://packages.debian.org/ja/sid/hello> が既に存在します

^{*9} git-buildpackage ではデフォルトのブランチ名は現状 master のままになっています。

git-buildpackage において git のブランチは以下の使い分けをします。

- master (または main) ブランチ：debian パッケージ全体の内容を含むブランチ
- upstream ブランチ：upstream の配布物をそのまま展開した内容を含むブランチ
- pristine-tar ブランチ：debian パッケージのビルドに使う upstream ブランチから生成する orig.tar.gz と upstream の tarball の差分をバイナリで保持するブランチ

gbp import-orig コマンドに `-pristine-tar` オプションを指定した場合は pristine-tar ブランチも作成されます。

pristine-tar ブランチの作成は必須ではありません。ただ、tarball を展開して upstream ブランチに保存して、逆に upstream ブランチのファイルから再生成した tarball ファイルのハッシュ値が、gbp import-orig で取り込んだ tarball のハッシュ値と同じにならないことがあります。pristine-tar ブランチにはその差分をバイナリで保存して、gbp buildpackage で debian パッケージをビルドするときに適用することでハッシュ値が同じになるように調整しています。

```
$ git checkout pristine-tar
Switched to branch 'pristine-tar'
Your branch is up to date with 'origin/pristine-tar'.

$ ls -l
合計 28
-rw-r--r-- 1 norimitu norimitu 6107  3月 19 04:25 hello_2.10.orig.tar.gz.delta
-rw-r--r-- 1 norimitu norimitu  41  3月 19 04:25 hello_2.10.orig.tar.gz.id

$ cat hello_2.11.orig.tar.gz.id
61732d00c5e503e9ced7f7f57f5b1c14a449c95c

$ file hello_2.11.orig.tar.gz.delta
hello_2.11.orig.tar.gz.delta: gzip compressed data, from Unix, original size modulo 2^32 30720

$ git checkout master
```

4.3.3 debian ディレクトリを生成する

debian パッケージを作成するのは、main ブランチに debian ディレクトリを作成します。dh.make コマンドを使って作成できます。

```
$ dh_make -p hello_2.10

Type of package: (single, indep, library, python)
[s/i/l/p]?
Maintainer Name   : Norimitsu Sugimoto
Email-Address     : dictoss@live.jp
Date              : Thu, 17 Mar 2022 13:27:56 +0000
Package Name      : hello
Version           : 2.10
License           : blank
Package Type      : single
Are the details correct? [Y/n/q]
Skipping creating ../hello_2.10.orig.tar.gz because it already exists
Done. Please edit the files in the debian/ subdirectory now.
```

dch コマンドで debian/changelog ファイルを開いて編集します^{*10}。

```
$ dch

hello (2.10-1) UNRELEASED; urgency=medium

 * Initial release (Closes: #nnnn) <nnnn is the bug number of your ITP>

-- Norimitsu Sugimoto <dictoss@live.jp> Thu, 17 Mar 2022 13:31:04 +0000
```

debian/control ファイルを編集し、パッケージのメタ情報を更新します。

^{*10} 通常、debian パッケージを新規作成する場合は ITP のバグ報告を行っている前提があり、debian/changelog ファイルにそのバグ番号を含めます。

```

$ vim debian/control

Source: hello
Section: devel
Priority: optional
Maintainer: Norimitsu Sugimoto <dictoss@live.jp>
Build-Depends: debhelper-compat (= 13), autotools-dev, gnulib, help2man
Standards-Version: 4.6.0
Homepage: http://www.gnu.org/software/hello/
#Vcs-Browser: https://salsa.debian.org/debian/hello
#Vcs-Git: https://salsa.debian.org/debian/hello.git
Rules-Requires-Root: no

Package: hello
Architecture: any
Depends: ${shlibs:Depends}, ${misc:Depends}
Description: example package based on GNU hello
 The GNU hello program produces a familiar, friendly greeting. It
 allows non-programmers to use a classic computer science tool which
 would otherwise be unavailable to them.
.
 Seriously, though: this is an example of how to do a Debian package.
 It is the Debian version of the GNU Project's `hello world' program
 (which is itself an example for the GNU Project).

```

debian/rules ファイルを編集し、debhelper を使って debian パッケージをビルドできるように調整します。

```

$ vim debian/rules

%:
    dh $@

override_dh_auto_clean:
    [ ! -f Makefile ] || $(MAKE) distclean

```

それでは gbp buildpackage コマンドを使って debian パッケージをビルドしてみます。しかし、コミットしていないファイルがあるとコマンドがエラーになります。まずは編集した debian ディレクトリを git へコミットします。

```

$ gbp buildpackage --git-pristine-tar --git-debian-branch=main

gbp:error: You have uncommitted changes in your source tree:
gbp:error: On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
  debian/

nothing added to commit but untracked files present (use "git add" to track)
gbp:error: Use --git-ignore-new to ignore.

```

```

$ git add debian
$ git commit -m "add debian directory."

```

再度、gbp buildpackage コマンドを実行します。debian パッケージのビルドに成功しました。しかし、lintian の警告がたくさん出ていますのでリリースするまでに警告をできるだけ減らすパッケージを見直す必要があります。

```

$ gbp buildpackage --git-pristine-tar --git-debian-branch=main

gbp:info: Performing the build
dpkg-buildpackage -us -uc -ui -i -I
dpkg-buildpackage: info: source package hello
dpkg-buildpackage: info: source version 2.10-1
dpkg-buildpackage: info: source distribution UNRELEASED
dpkg-buildpackage: info: source changed by Norimitsu Sugimoto <dictoss@live.jp>
dpkg-source -i -I --before-build .
dpkg-buildpackage: info: host architecture amd64
debian/rules clean

(省略)

dpkg-buildpackage: info: full upload (original source is included)
Now running lintian hello_2.10-1_amd64.changes ...
E: hello: changelog-is-dh_make-template
(省略)
W: hello: wrong-bug-number-in-closes #nnnn in the installed changelog (line 4)
Finished running lintian.

```

これで debian パッケージのビルドができました。ビルド時に生成されたファイルを削除してから、main ブランチにタグを作成し、リモートの git リポジトリに push します。

```
$ gbp buildpackage --git-tag-only --git-debian-branch=main
gbp:info: Tagging Debian package 2.10-1 as debian/2.10-1 in git

$ git tag
debian/2.10-1
upstream/2.10
```

```
$ git remote add origin git@salsa.debian.org:dictoss-guest/hello.git
$ git remote -v
origin git@salsa.debian.org:dictoss-guest/hello.git (fetch)
origin git@salsa.debian.org:dictoss-guest/hello.git (push)

$ git push -u origin --all
$ git push -u origin --tag
```

4.4 upstream で新しいバージョンの tarball がリリースされた場合の作業

まず、更新された tarball をダウンロードします。

```
$ cd ..
$ wget https://ftp.gnu.org/gnu/hello/hello-2.11.tar.gz
```

gbp import-orig コマンドで tarball を git のローカルブランチに取り込みます。

```
$ gbp import-orig --debian-branch=main --pristine-tar ../hello-2.11.tar.gz

What is the upstream version? [2.11]
gbp:info: Importing '../hello-2.11.tar.gz' to branch 'upstream'...
gbp:info: Source package is hello
gbp:info: Upstream version is 2.11
gbp:info: Replacing upstream source on 'main'
gbp:info: Successfully imported version 2.11 of ../hello_2.11.orig.tar.gz
```

git branch を実行するとブランチの数に変化はありません。

```
$ git branch
* main
  pristine-tar
  upstream
```

git tag を実行すると "upstream/2.11" が追加されています。

```
$ git tag
debian/2.10-1
upstream/2.10
upstream/2.11
```

dch コマンドで debian/changelog ファイルを開いて編集します。

```
$ dch

hello (2.11-1) UNRELEASED; urgency=medium

 * New upstream version 2.11

-- Norimitsu Sugimoto <dictoss@live.jp> Fri, 19 Mar 2022 14:10:38 +0000

hello (2.10-1) UNRELEASED; urgency=medium

 * Initial release (Closes: #nnnn) <nnnn is the bug number of your ITP>

-- Norimitsu Sugimoto <dictoss@live.jp> Fri, 18 Mar 2022 13:39:37 +0000
```

debian ディレクトリの変更内容をコミットします。

```
$ git add debian
$ git commit -m "New upstream version 2.11."
```

gbp buildpackage を実行して debian パッケージをビルドしてみるとエラーになりました。

```

$ gbp buildpackage --git-pristine-tar --git-debian-branch=main

(省略)
*** error: gettext infrastructure mismatch: using a Makefile.in.in from gettext
version 0.18 but the autoconf macros are from gettext version 0.20
make[3]: *** [Makefile:165: check-macro-version] Error 1
make[3]: *** Waiting for unfinished jobs...
(省略)
dh_auto_build: error: make -j3 returned exit code 2
make: *** [debian/rules:18: binary] エラー 25
dpkg-buildpackage: error: debian/rules binary subprocess returned exit status 2
debuild: fatal error at line 1182:
dpkg-buildpackage -us -uc -ui -i -I failed
gbp:error: 'debuild -i -I' failed: it exited with 29

```

gettext のバージョンが sid のバージョンと tarball のファイルに書いてあるバージョンが合わないためエラーになったようです。gbp pq コマンドを使ってパッチ作成用のブランチを作ります。gbp pq import コマンドを実行すると patch-queue/main ブランチが作成されチェックアウトした状態になります。

```

$ gbp pq import
gbp:info: Trying to apply patches at 'cacbfce059a86c836513ff70eb6af780d9255910'
gbp:info: 0 patches listed in 'debian/patches/series' imported on 'patch-queue/main'

$ git branch
main
* patch-queue/main
pristine-tar
upstream

```

patch-queue/main ブランチ上で変更が必要なファイルを編集します。

```

$ vim configure.ac
$ git diff
diff --git a/configure.ac b/configure.ac
index 6d52c41..f101d7b 100644
--- a/configure.ac
+++ b/configure.ac
@@ -62,7 +62,7 @@ dnl Ensure VLAs are not used.
AC_DEFINE([GNULIB_NO_VLA], [1], [Define to 1 to disable use of VLAs])

dnl i18n support from GNU gettext.
-AM_GNU_GETTEXT_VERSION([0.18.1])
+AM_GNU_GETTEXT_VERSION([0.21])
AM_GNU_GETTEXT([external])

AC_CONFIG_FILES([Makefile

```

patch-queue/main ブランチに変更したファイルをコミットします。

```

$ git add configure.ac
$ git commit -m "change to gettext version to use with sid."
[patch-queue/main cc568aa] change to gettext version to use with sid.
1 file changed, 1 insertion(+), 1 deletion(-)

```

すべてのファイルの変更が完了した後、gbp pq export コマンドを実行して編集を終了します。gbp pq export コマンドを実行すると main ブランチをチェックアウトした状態になり、main ブランチと patch-queue/main ブランチの差分が debian/patches/ ディレクトリに生成されます。

```

$ gbp pq export
gbp:info: On 'patch-queue/main', switching to 'main'
gbp:info: Generating patches from git (main..patch-queue/main)

```

```

$ git branch
* main
patch-queue/main
pristine-tar
upstream

```

```

$ git status .
On branch main
Your branch is ahead of 'origin/main' by 3 commits.
(use "git push" to publish your local commits)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  debian/patches/

nothing added to commit but untracked files present (use "git add" to track)

$ ls -l debian/patches/
合計 8
-rw-r--r-- 1 norimitu norimitu 596  3月 19 02:28 0001-change-to-gettext-version-to-use-with-sid.patch
-rw-r--r-- 1 norimitu norimitu  53  3月 19 02:28 series

```

生成された `debian/patches` ディレクトリを `main` ブランチにコミットします。

```

$ git add debian/patches
$ git commit -m "change to gettext version to use with sid."
[main 6ae3e38] change to gettext version to use with sid.
 2 files changed, 22 insertions(+)
 create mode 100644 debian/patches/0001-change-to-gettext-version-to-use-with-sid.patch
 create mode 100644 debian/patches/series

```

`gbp buildpackage` を実行すると、更新した `debian` パッケージをビルドできます。

```

$ gbp buildpackage --git-pristine-tar --git-debian-branch=main

$ ls ..
hello                hello_2.10-1.debian.tar.xz      hello_2.10-1_amd64.deb          hello_2.11-1_amd64.buildinfo
hello-2.10.tar.gz    hello_2.10-1.dsc                hello_2.10.orig.tar.gz        hello_2.11-1_amd64.changes
hello-2.11.tar.gz    hello_2.10-1_amd64.build        hello_2.11-1.debian.tar.xz    hello_2.11-1_amd64.deb
hello-dbgSYM_2.10-1_amd64.deb  hello_2.10-1_amd64.buildinfo  hello_2.11-1.dsc              hello_2.11.orig.tar.gz
hello-dbgSYM_2.11-1_amd64.deb  hello_2.10-1_amd64.changes     hello_2.11-1_amd64.build

```

ビルド時に生成されたファイルを削除してから、`main` ブランチにタグを作成し、リモートの `git` リポジトリに `push` します。

```

$ gbp buildpackage --git-tag-only --git-debian-branch=main
gbp:info: Tagging Debian package 2.11-1 as debian/2.11-1 in git

$ git tag
debian/2.10-1
debian/2.11-1
upstream/2.10
upstream/2.11

```

```

$ git push -u origin main upstream pristine-tar
$ git push -u origin --tag

```

作業を終えた `patch-queue/main` ブランチは不要のため削除します。

```

$ git branch -D patch-queue/main

```

4.5 今回取り扱えなかったこと

今回取り扱えなかったことはたくさんあります。

- `upstream` が `git` リポジトリのみで `tarball` を提供していない場合
- `sid`、`stable`、`stable-backports`、`experimental` を平行で管理する方法
- `gbp buildpackage -git-pbuilder`
- `gbp.conf`
- `upstream` のソースコードにすでに `debian` ディレクトリが存在している場合

4.6 おわりに

git-buildpackage パッケージに入っている gbp コマンドを調べてみました。quilt コマンドを使ったパッチの作成を gbp pq コマンドでより便利に使えるのがよいと思いました。

実際にパッケージをメンテナンスするときには、複数のリリースを 1 つの git リポジトリで管理したいのだと思います。その場合にどのように git-buildpackage を使いこなせばよいか、まだまだ勉強が必要であると感じました。

4.7 参考文献

- Debian Wiki - PackagingWithGit <https://wiki.debian.org/PackagingWithGit>
- Debian パッケージング道場 git-buildpackage
<https://tokyodebian-team.pages.debian.net/pdf2015/debianmeetingresume201509-presentation.pdf>
- Debian 新メンテナガイド 第 6 章 パッケージのビルド
<https://www.debian.org/doc/manuals/maint-guide/build.ja.html>
- git-buildpackage を用いた deb パッケージ管理方法の紹介
<https://blog.cybozu.io/entry/2019/04/11/110000>

5 DDTP 及び DDTSS の紹介

杉本 典充

本資料は発表時のプレゼンテーションから一部機械的にフォーマットを変更しています。読みにくい部分をご容赦ください。

5.1 DDTP

- The Debian Description Translation Project
- Debian パッケージの説明文の翻訳を取り組むプロジェクト
- <https://www.debian.org/international/l10n/ddtp.ja.html>
- 翻訳状況の統計情報、翻訳作業の支援ツールを提供
- Debian パッケージの翻訳を行うと以下の場所などに翻訳文が表示される
 - <https://packages.debian.org/ja/>
 - `apt show packagename`

5.2 DDTP

- パッケージ説明文翻訳の言語別の進捗
 - <https://ddtp.debian.org/stats/stats-sid.html>
 - Required、Important、Standard、Optional、Extra の区分は Debian パッケージの優先度^{*11}
 - * Required、Important、Standard のパッケージは利用者が多いため優先的に翻訳するとよい
 - Popcon500、PopconRank は popularity-contest^{*12} パッケージで収集されたパッケージの利用状況のランキング
 - * Popcon500 のパッケージは利用者が多いため優先的に翻訳するとよい
 - 表のセルにマウスを乗せて少し待つとツールチップが表示され、未翻訳のパッケージ名が表示される

5.3 DDTSS

- Debian パッケージの説明文の翻訳とレビューを行う Web サイト
 - Debian Distributed Translation Server Satellite
 - <https://ddtp.debian.org/ddtss/index.cgi/xx>
 - sid のパッケージが作業対象

^{*11} <https://www.debian.org/doc/manuals/debian-faq/pkg-basics.ja.html#priority>

^{*12} <https://popcon.debian.org/>

- 翻訳を行い、3人のレビューがOKになると反映される
- アカウントの登録が必要

5.4 そのほかの翻訳ツール Weblate

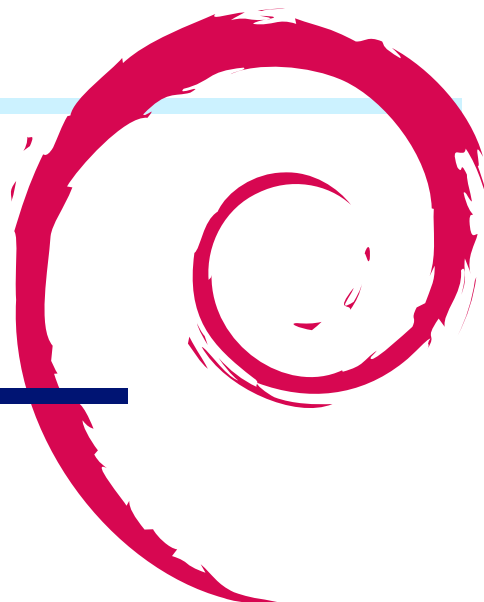
- プロジェクト横断でドキュメントの翻訳作業を支援する Web サイト
- <https://hosted.weblate.org/>
- Debian では以下のドキュメントの翻訳作業が行える
 - <https://hosted.weblate.org/projects/debian-installer/>
 - <https://hosted.weblate.org/projects/debian-installation-guide/>
 - <https://hosted.weblate.org/projects/debian-reference/>
 - <https://hosted.weblate.org/projects/debian-handbook/>

5.5 まとめ

- DDTP の取り組みを紹介しました
- DDTSS の Web サイト <https://ddtp.debian.org/ddtss/index.cgi/xx>
- 2023 年に bookworm をリリースするまでにできるだけ翻訳作業を進めましょう

6 アプリにおける多言語対応の仕組み

杉本 典充



6.1 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。

6.2 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^{*13}。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^{*14}): 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^{*15}): 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 リファレンスに「8. 国際化と翻訳」の記事があります^{*16}。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。

6.3 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。

^{*13} <https://www.debian.org/doc/manuals/debian-reference/ch08.ja.html>

^{*14} 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。

^{*15} localization のこと。文字数が長いため簡略した記述として使われる。

^{*16} <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^{*17}も存在します。

6.4 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

6.5 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

6.5.1 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。

^{*17} LANG=C と書いた方が馴染みがあるかもしれませんが。

```

#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 ppointja pomergeja poininstallja
all: $(PROG)
$(PROG): $(OBJS) $(PO_OBJS)
    $(CC) $(LDFLAGS) -o $@ $(LIBS) $(OBJS)
clean:
    $(RM) $(OBJS) $(PO_OBJS) *~
pot:
    xgettext -k "_" -o $(PROG).pot $(SRCS)
ppointja:
    msginit --locale ja_JP.UTF-8 --input $myhello.pot --output-file po/ja.po
pomergeja:
    msgmerge -U po/ja.po myhello.pot
poininstallja:
    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 $@ $<

```

6.5.2 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^{*18}
 - 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.
```

6.5.3 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。

^{*18} `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*¹⁹。

```

#: 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

*¹⁹ vi など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*20。

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。
```

6.5.4 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。"
```

6.6 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。

6.7 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>

*20 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。

7 Gmail とパスワード

yy-y-ja-jp



7.1 Debian とメール

Debian でメールを使う、特に送る機会としては次のような場面が挙げられます。

- キーサイン*21
- バグ報告*22
- 投票 (Debian 公式開発者の場合)
- ...

昨今通常のメールでは Web メールを使うことも多いかと思いますが、こういった場面では Web メール以外を使うこともあったりします。

7.2 Google 曰く “安全性の低いアプリ”

特に Debian など FLOSS の活動をするためにフリーメールアドレスとして Google 社の Gmail を使っている人も多いでしょう。Gmail は Web メールとして使う分には長らく大きく変わっていないのですが、最近 Web メール以外のメールクライアントからの利用がセキュリティを理由に変更・制限されてきています。

Gmail は最近 (2022 年 2 月) までメールクライアントから Google アカウントのユーザー名とパスワードでアクセスする設定ができていました。しかしそれは「安全性の低いアプリ」とみなされ、(執筆時) 2022 年 5 月現在は新たな有効化ができません。私が使っている有効化済みアカウントで設定画面を見てみます。

(以下引用元: Google 「セキュリティ」 <https://myaccount.google.com/security?hl=ja> 2022 年 5 月 17 日閲覧)



2022 年 5 月 30 日より、この設定は利用できなくなります。

*21 コマンドラインツール `caff` (signing-party パッケージ) がよく使われますが、MTA (Mail Transfer Agent) の設定が必要です。

*22 なお端末から Debian へバグ報告できるツール `reportbug` は、最近では初回実行時に名前とメールアドレスさえ入力すれば MTA の設定なしにバグ報告できます。

(引用ここまで)
とあります。

7.3 “安全性の低いアプリと Google アカウント”

(以下引用元: Google「安全性の低いアプリと Google アカウント - Google アカウント ヘルプ」
<https://support.google.com/accounts/answer/6010255?hl=ja> 2022 年 5 月 17 日閲覧)

安全性の低いアプリと Google アカウント

アカウントを安全に保つため、**2022 年 5 月 30 日**より、Google は、ユーザー名とパスワードのみで Google アカウントにログインするサードパーティ製のアプリとデバイスについてサポートを終了いたします。

この期限は、Google Workspace または Google Cloud Identity のお客様には適用されません。これらのお客様への適用日は、Workspace のブログで後日お知らせいたします。

詳しくは、下記のとおりです。

Apple デバイスでのログインに関する注意事項。Google アカウントにユーザー名とパスワードのみを使用して最近ログインしていない場合は、**2022 年 2 月 28 日**以降、Google アカウントの種類を使用する新しいログイン操作しか行えません。最近ログインした場合は、**2022 年 5 月 30 日**まで、ユーザー名とパスワードを使用して Google アカウントに引き続きログインできます。

アカウントを安全に保つため、2022 年 5 月 30 日より、Google は、ユーザー名とパスワードのみで Google アカウントにログインするサードパーティ製のアプリとデバイスについてサポートを終了いたします。

(引用ここまで)

今後 Web メール以外のメールクライアントを使う場合は何らかの別の方法で Gmail へアクセスする必要があります。

7.4 Google アカウント “アプリ パスワード”

その別の方法の一つとして Google アカウントには“アプリ パスワード”というものが用意されています。

(以下引用元: Google「アプリ パスワードでログインする - Google アカウント ヘルプ」
<https://support.google.com/accounts/answer/185833?hl=ja> 2022 年 5 月 17 日閲覧)

ヒント: アプリ パスワードは推奨されておらず、ほとんどの場合は不要です。アカウントの安全性を保つには、「Google でログイン」機能を使用してアプリを Google アカウントに接続します。

アプリ パスワードとは、安全性の低いアプリやデバイスに Google アカウントへのアクセスを許可する 16 桁のパスコードです。アプリ パスワードは 2 段階認証プロセスを有効にしているアカウントでのみ使用できます。

(引用ここまで)

“アプリ パスワード”とはアカウントパスワードのアプリ向け代替です。しかし Google は推奨していません。Google は代わりに「Google でログイン」機能 (OAuth 認証) というものを推奨しています。ただしこれはメールクライアントが対応している必要があるため、ご利用のメールクライアントによっては難しいでしょう。例えば Sylpheed が OAuth 認証に対応しておらず、フォーク版の Claws Mail は対応している、といった具合に対応できているものはよいのですが、昔ながらのメールクライアントやツール、MTA では厳しいです (一部の MTA はかなり頑張ればできますが...)。そこでここではアプリ パスワードを使ってみます。

アプリ パスワードを Google アカウントで使うには“2 段階認証プロセス”の有効化が必須になっているようです。以下これを見ていきます。

7.5 Google アカウント “2 段階認証プロセス”

(以下引用元: Google 「2 段階認証プロセスを有効にする - パソコン - Google アカウント ヘルプ」
<https://support.google.com/accounts/answer/185839> 2022 年 5 月 17 日閲覧)

2 段階認証プロセス (2 要素認証プロセスとも呼ばれます) は、パスワードが盗まれた場合に備えてアカウントのセキュリティを強化するものです。2 段階認証プロセスを設定すると、アカウントへのログインは、次のものを使って 2 段階で行うことになります。

- 自分が把握している情報 (パスワードなど)
- 自分が持っているもの (お使いのスマートフォンなど)

(引用ここまで)

基本的には自分の身近にあるものも使って本人を認証するものです。もはや電話を肌身離さず持っている人も多いかとは思いますが、PC の作業中に電話を求められるのはちょっと微妙な気もしますが...

さて、自分の Google アカウントで 2 段階認証プロセスを有効化してみます。有効化するまで「アプリ パスワード」機能はどの画面にも出てこないようです。

Google アカウントの「セキュリティ」画面で「2 段階認証プロセス」というメニューを進めていくと次のような画面になりました。

(以下引用元: Google 「2 段階認証プロセス」

<https://myaccount.google.com/signinoptions/two-step-verification> 2022 年 5 月 17 日閲覧)

← 2 段階認証プロセス

電話番号の設定

使用する電話番号を選択してください。

Google はこの番号をアカウントのセキュリティ保護にのみ使用します。
Google Voice 番号は使用しないでください。
データ通信料がかかる場合があります。

コードの取得方法

テキストメッセージ 音声通話

他のオプションを表示

手順 1 / 3 次へ

(引用ここまで)

「自分が持っているもの」として選べるのは電話がデフォルトのようです。

他の方法も見てみますが...

(以下引用元: Google 「2 段階認証プロセス」

<https://myaccount.google.com/signinoptions/two-step-verification> 2022 年 5 月 17 日閲覧)

← 2段階認証プロセス

The screenshot shows the '電話番号の設定' (Set phone number) screen. At the top, there are icons for a lock, a person, a smartphone, a phone, and a speech bubble. Below the icons, the title '電話番号の設定' is followed by the instruction '使用する電話番号を選択してください。' (Select the phone number you want to use). There is a dropdown menu with a single option. Below this, a note states: 'Googleはこの番号をアカウントのセキュリティ保護にのみ使用します。Google Voice 番号は使用しないでください。データ通信料がかかる場合があります。' (Google uses this number only for account security protection. Do not use Google Voice numbers. Data charges may apply). Under 'コードの取得方法' (How to get the code), there are two radio buttons: 'テキスト メッセージ' (Text message) which is selected, and '音声通話' (Voice call). A '他のオプションを表示' (Show other options) link is visible, which has opened a dropdown menu listing 'セキュリティ キー' (Security key) and 'Google からのメッセージ' (Message from Google).

(引用ここまで)

- テキスト メッセージ・音声通話 – 要はどちらも電話
- セキュリティ キー – 自分が持っているハードウェアキー
- “Google からのメッセージ” – スマートフォンで受信するしかない(電話)

ということで結局電話が現状ほとんどの人が持っていないだろうハードウェアキーしか選べなさそうです。

あきらめてここでは音声通話にしてみました。電話番号を入力すると Google から電話が掛かってくるので、自動音声聞いて確認コードを入力します。完了すると 2 段階認証プロセスを有効にできるようになります。

有効化すると、Google アカウントの「セキュリティ」画面で「アプリ パスワード」メニューが出現します。

7.6 Google アカウント アプリ パスワード

その「アプリ パスワード」メニューを進めてみます。

(以下引用元: Google 「アプリ パスワード」<https://myaccount.google.com/apppasswords> 2022 年 5 月 17 日閲覧)

← アプリ パスワード

アプリパスワードを使用すると、2段階認証プロセスに対応していないデバイス上のアプリから Google アカウントにログインできるようになります。このパスワードは一度入力すれば、以降は覚えておく必要はありません。詳細

The screenshot shows the 'Generate app passwords' screen. It starts with the message 'アプリパスワードがありません。' (You don't have app passwords). Below that is the instruction 'アプリパスワードを生成するアプリとデバイスを選択してください。' (Select the app and device you want to generate app passwords for). There are two dropdown menus: 'アプリを選択' (Select app) and 'デバイスを選択' (Select device). A '生成' (Generate) button is located at the bottom right.

このパスワードは一度入力すれば、以降は覚えておく必要はありません。

← アプリパスワード

アプリパスワードを使用すると、2段階認証プロセスに対応していないデバイス上のアプリから Google アカウントにログインできるようになります。このパスワードは一度入力すれば、以降は覚えておく必要はありません。詳細

アプリパスワードがありません。

アプリパスワードを生成するアプリとデバイスを選択してください。

メール ▼ デバイスを選択

- iPhone
- iPad
- BlackBerry
- Mac
- Windows Phone
- Windows パソコン
- その他 (名前を入力)

生成

(引用ここまで)

アプリとデバイスを選択もしくは入力することになっていますが、特に認証自体には影響なく何を入れてもいいようです。

生成すると次のようになりました。

(以下引用元: Google「アプリパスワード」<https://myaccount.google.com/apppasswords> 2022年5月18日閲覧)

生成されたアプリパスワード

お使いのデバイスのアプリパスワード

wupq hoix blxl eifk

使い方

設定しようとしているアプリケーションまたはデバイスの Google アカウントの設定画面を開きます。パスワードを上に表示されている 16 文字のパスワードに置き換えます。

このアプリパスワードは、通常のパスワードと同様に Google アカウントへの完全なアクセス権が付与されます。このパスワードを覚えておく必要はないので、メモしたり誰かと共有したりしないでください。

完了

Email
securesally@gmail.com

Password
●●●●●●●●●●●●●●●●

(引用ここまで)

このアプリパスワードをメールクライアントでアカウントパスワードの代わりに保存して使います。なおこの画面を閉じると、保存したはずだということでもう見られませんが、

今のところ Web で Google アカウントにログインするときに 2 段階認証になり面倒になるだけで、ほかの使い勝手は基本的に変わらないはずですが、いつまでこの非推奨というアプリパスワードが使えるのかは不明ですが...

なお、この後で 2 段階認証をオフにすると、このアプリパスワードは残念ながら即使えなくなります。

7.7 Google アカウント 2 段階認証プロセス

さて、ここまで標準的な方法を見てきました。なんとか電話（やハードウェアキー）以外の方法も使えないか見てみます。

（以下引用元: Google「2 段階認証プロセス」

<https://myaccount.google.com/signinoptions/two-step-verification> 2022 年 5 月 17 日閲覧)

2 つ目の手順を追加してログインを確認する

追加のバックアップ手順をセットアップして、別の手順が使用できない時でもログインできるようにします。

	バックアップ コード この印刷用の 1 回限りのパスコードを利用すると、旅行中などスマートフォンが手元にないときにもログインすることができます。	>
	Google からのメッセージ Google からのメッセージを受信するには、スマートフォンで Google アカウントにログインするだけです。 新しいデバイスでパスワードを入力すると、ログイン済みのすべてのスマートフォンに Google からメッセージが送信されます。そのうちのいずれかをタップして確認します。 メッセージに対応しているどのデバイスにも現在ログインしていません。	>
	認証システム アプリ スマートフォンがオフラインの場合でも、認証システム アプリを使って確認コードを料金なしで取得できます。Android と iPhone に対応しています。	>
	セキュリティ キー セキュリティ キーは、安全にログインできるようにする認証方法です。スマートフォンの組み込みのキー、Bluetooth を介して使用するキー、パソコンの USB ポートに直接挿すキーがあります。	>

2 つ目の手順が不要なデバイス

お使いのパソコンのような信頼できるデバイスでは 2 つ目の手順をスキップできます。

（引用ここまで）

2 段階認証プロセスを有効にすると、電話・ハードウェアキー以外の選択肢が増えることがわかります。

- バックアップコード – これは普段使いしてはダメで印刷保管しておくものです。
- 認証システム アプリ – こちらを以下見ていきます。

7.8 Google アカウント “認証システム アプリ”

先ほどの画面で

（以下引用元: Google「2 段階認証プロセス」

<https://myaccount.google.com/signinoptions/two-step-verification> 2022 年 5 月 17 日閲覧)

認証システム アプリ

スマートフォンがオフラインの場合でも、認証システム アプリを使って確認コードを料金なしで取得できます。Android と iPhone に対応しています。

（引用ここまで）

とありましたが、実はスマートフォンでなくてもできたりするのでやってみます。

(以下引用元: Google「認証システム アプリ」)

<https://myaccount.google.com/two-step-verification/authenticator> 2022年5月21日閲覧)

← 認証システム アプリ

テキストメッセージを受信する代わりに、認証システムアプリから確認コードを取得します。スマートフォンがオフラインでも利用できます。

まず、Google 認証システムを Google Play ストアまたは iOS App Store からダウンロードします。



+ 認証システムを設定

まず、Google 認証システムを Google Play ストアまたは iOS App Store からダウンロードします。

(引用ここまで)

ここで“認証システム アプリ”は“Google 認証システム (Google Authenticator)”アプリのことを指してはいます。

7.9 Google アカウント 認証システム アプリの設定 (1)

“認証システムを設定”を進めると QR コードが表示されます。

(以下引用元: Google「認証システム アプリ」)

<https://myaccount.google.com/two-step-verification/authenticator> 2022年5月18日閲覧)

認証システム アプリの設定

- Google 認証システム アプリで [+] をタップします
- [QR コードをスキャン] を選択します

スキャンできない場合

(引用ここまで)

などと書いてありますが、とりあえず無視して表示されている QR コード画像を安全な場所に保存してみることにします。

また、“スキャンできない場合”というリンクも見てみます。

(以下引用元: Google「認証システム アプリ」)

<https://myaccount.google.com/two-step-verification/authenticator> 2022年5月18日閲覧)

1. Google 認証システム アプリで、[+] > [セットアップ キーを入力] をタップします。
2. メールアドレスと以下のキーを入力します (空白は無視されます):
nfiw xjug uqtt ewx2 er5v zuxm mbgh 3jt3
3. [時間ベース] が選択されていることを確認します
4. [追加] をタップして終了します

(引用ここまで)

QR コードの代わりに“セットアップ キー”が表示されています。以下これらを見ていきます。

7.10 QRコードと“セットアップキー”

先ほど保存してみた QR コード画像（ここでは index.png）を解析してみます（以下私のメールアドレス部分は一部隠しています）。QR コードの解析用コマンドラインツール zbarimg（zbar-tools パッケージ）を使ってみます。

```
# apt install zbar-tools
#
$ zbarimg index.png
QR-Code:otpauth://totp/Google%3A*****%40gmail.com?secret=nfiwxjuguqttewx2er5vzuxmmbgh3jt3&issuer=Google
scanned 1 barcode symbols from 1 images in 0.01 seconds
$
$ zbarimg -q --raw index.png | sed 's/.*secret=\([[:alnum:]]*\).*\/1/'
nfiwxjuguqttewx2er5vzuxmmbgh3jt3
$
```

認証システム アプリに必要な鍵情報が URI 形式で書かれています。この URI 形式自体は標準化されているわけではないようですが、Google 以外の他サービスも概ね“Google 認証システム”アプリに合わせているようです。パーセントエンコードをデコードした各パラメータは次のようになっているようです。

- totp – “時間ベース” (Time-based One-Time Password) であることを示しています
- *****@gmail.com – “メールアドレス” アカウント名
- nfiwxjuguqttewx2er5vzuxmmbgh3jt3 – “セットアップキー” 秘密鍵 (base32 エンコード)
- Google – サービス名

QR コードには同じ“セットアップキー”が含まれていることがわかります。また、“セットアップキー”を手入力する場合とは異なりメールアドレスも含まれているようです。

7.11 互換“認証システム アプリ”

さてこの“認証システム アプリ”ですが、“時間ベース” – TOTP (Time-based One-Time Password)^{*23} を実装しているソフトウェアなら互換性があります。Google 社製のほかに例えば Microsoft 社の Microsoft Authenticator アプリなどもあります。

なお確認コードを表示（生成）するには先ほどの秘密鍵“セットアップキー”は必要ですが、他のサービス名やアカウント名といったものは不要です。

互換アプリのなかには、以下のように直近の確認コードだけでなく次の番号を一気に出力する機能も持った Debian でインストールできるコマンドラインツール oathtool があります。

```
# apt install oathtool
#
$ zbarimg -q --raw index.png | sed 's/.*secret=\([[:alnum:]]*\).*\/1/' | oathtool --totp --base32 -
789613
$ zbarimg -q --raw index.png | sed 's/.*secret=\([[:alnum:]]*\).*\/1/' | oathtool -w 10 --totp --base32 -
789613
067410
818083
260090
232806
105748
078883
190238
164080
397309
870025
$
```

これで電話を使わなくても Debian PC だけで完結できそうですね。ちなみに TOTP は時計（この例では Debian PC の時計）が狂っていると合わなくなります。

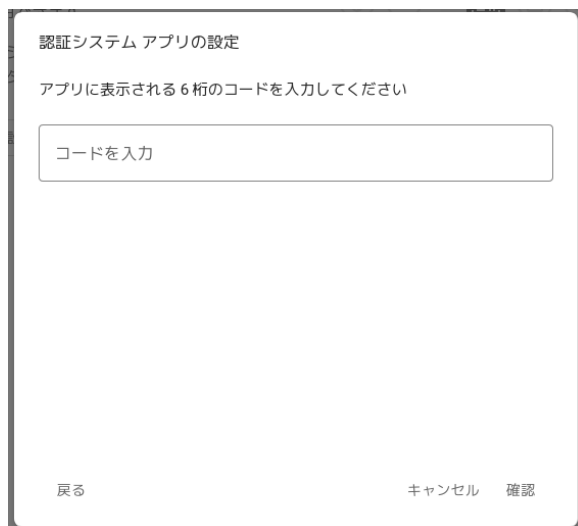
^{*23} RFC6238 “TOTP: Time-Based One-Time Password Algorithm” <https://www.ietf.org/rfc/rfc6238.txt>

7.12 Google アカウント 認証システム アプリの設定 (2)

さて、この互換アプリを使って Google アカウントを設定してみます。

(以下引用元: Google「認証システム アプリ」)

<https://myaccount.google.com/two-step-verification/authenticator> 2022 年 5 月 18 日閲覧)



アプリに表示される 6 桁のコードを入力してください

(引用ここまで)

ここで (互換) アプリの出力するコードを入力してみます。

(以下引用元: Google「認証システム アプリ」)

<https://myaccount.google.com/two-step-verification/authenticator> 2022 年 5 月 18 日閲覧)

← 認証システム アプリ

テキスト メッセージを受信する代わりに、認証システム アプリから確認コードを取得します。スマートフォンがオフラインでも利用できます。

まず、Google 認証システムを Google Play ストアまたは iOS App Store からダウンロードします。



(引用ここまで)

無事成功したようです。

この段階で「2 段階認証プロセス」画面に戻ると、最初に設定した電話が外せるようになっていました^{*24}。

^{*24} ちなみに本記事を執筆するために 2 段階認証付け外ししたところ Google に (少なくとも半日以上は) 電話番号ブロックされてしまったのでご注意ください。

7.13 認証システム アプリの鍵情報の管理

さて、この“認証システム アプリ”の鍵をどう管理するかがとても重要になります。いくつか方法があるでしょう。

- 認証システム アプリに管理してもらう（これが一般的な方法）
この方法では、アプリによってはコード生成に必須ではないアカウント名・サービス名も保存されます。可能性は低いとは思いますが、スマートフォンやアプリの脆弱性を突かれる、もしくはスマートフォンを盗まれてロックを突破されると、残るパスワードを突破されれば悪用されます。
- QR コード画像の保存
この方法では、アカウント名・サービス名もセットで保存することになるので、盗まれたときは残るパスワードさえ突破されれば悪用されます。そして保存場所がとても重要で、暗号化ファイルシステム等も考えられます。
- セットアップ キー自体をパスワードマネージャ等に保存
この方法なら不要な情報を除き最小限の秘密を保存できます。ただし、利用するパスワードマネージャ等自体の管理方法（保存場所等）がとても重要になります。

7.14 まとめ

メールクライアントで Gmail のアカウントパスワードが使えなくなりました。今後は OAuth 認証を使うよう Google は推奨していますが、メールクライアントの対応が必須です。対応していないメールクライアント・MTA は今まで使っていたアカウントパスワードの代わりにアプリ パスワード（Google は非推奨）を使うことで、現状特に使い勝手変わらず利用できます。ただしその Google アカウントに Web でログインする際に 2 段階認証プロセスが必須となっています。認証システム アプリ (Authenticator) 互換ソフトウェアを使えば 2 段階認証プロセスを例えば Debian PC でもできるようにはなります。ただし認証システム アプリの鍵情報の管理に気をつける必要があります。

7.15 参考文献

- 「第 508 回 Ubuntu でコマンドラインからワンタイムパスワードを扱う — gihyo.jp」
<https://gihyo.jp/admin/serial/01/ubuntu-recipe/0508>
- “Key Uri Format · google/google-authenticator Wiki”
<https://github.com/google/google-authenticator/wiki/Key-Uri-Format>

本資料のライセンスについて

本資料はフリー・ソフトウェアです。あなたは、Free Software Foundation が公表した GNU GENERAL PUBLIC LICENSE の "バージョン 2" もしくはそれ以降が定める条項に従って本プログラムを再頒布または変更することができます。

本プログラムは有用とは思いますが、頒布にあたっては、市場性及び特定目的適合性についての暗黙の保証を含めて、いかなる保証も行ないません。詳細については GNU GENERAL PUBLIC LICENSE をお読みください。

ソースコードについて

本資料のソースコードは Git を使って <https://salsa.debian.org/tokyodebian-team/monthly-report.git> からダウンロードできます。以下に方法を示します。

```
$ sudo apt update
$ sudo apt install git
$ git clone https://salsa.debian.org/tokyodebian-team/monthly-report.git
```

GNU GENERAL PUBLIC LICENSE Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's

source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to

control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>
```

```
This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.
```

```
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.
```

```
You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
`Gnomovision' (which makes passes at compilers) written by James Hacker.
```

```
<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Debian オープンユースロゴライセンス

Copyright (c) 1999 Software in the Public Interest
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be

included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

『あんどきゅめんでっど でびあん』について

本書は、東京および関西周辺で毎月行なわれている『東京エリア Debian 勉強会』および『関西 Debian 勉強会』で使用された資料・小ネタ・必殺技などを一冊にまとめたものです。収録範囲は 2021/12 ~ 2022/05 まで。

内容は無保証、つっこみなどがあれば勉強会にて。



あんどきゅめんでっど でびあん 2022 年冬号

2022 年 12 月 31 日 初版第 1 刷発行

東京エリア Debian 勉強会/関西 Debian 勉強会 (編集・印刷・発行)

<https://debianjp.connpass.com>

<https://tokyodebian-team.pages.debian.net>

<https://wiki.debian.org/KansaiDebianMeeting>

