Grand Unified Debian

銀河系唯一のDebian専門誌

東京エリア/関西Ｄｅｂｉａｎ勉強会

あんどきゅめんてっど でびあん 2024 年冬号 2024 年 12 月 30 日 初版発行

デビアン勉強会

目次

1	Introduction	2
2	Debian Updates	3
3	WSL2 での sid 生活	10
4	autopkgtest について調べてみた	16
5	General Resolution:tag2upload は Debian にどんな変化をもたらすのか	22
6	より新しい Mozc を Debian でも使いたい	28

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 をがりがりと作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
- 定期的に集まれる場所
- 資料の作成

それでは、楽しい一時をお楽しみ下さい。

2 Debian Updates

Norimitsu SUGIMOTO (杉本 典充)

2.1 Debian とは？

- `https://www.debian.org/`
- ボランティアでフリー/オープンでユニバーサルな「オペレーティングシステム (OS) 」を開発する「コミュニティ」

2.2 Debian と OS

Debian はボランティアのみで開発しています

ディストリ	企業	ボランティア
Fedora	Red Hat 支援あり	あり
CentOS Stream	Red Hat 支援あり	あり
RHEL*1	Red Hat	なし
Debian	なし	あり
Ubuntu	Canonical	あり
openSUSE	SUSE 支援あり	あり
SLES*2	SUSE	なし

2.3 Debian というオペレーティングシステム

- 様々な用途に使える汎用的な作り
  - ノート PC、デスクトップ PC などの普段利用するコンピュータの OS

  - Linux サーバの OS
    * web サーバのシェア (2024-10-19 時点)[*3]：
      Linux サーバの内、Ubuntu 19.2 %、Debian 10.4 %、CentOS 3.9 %、Red Hat 0.3 %、AlmaLinux 0.3 %、...、Unknown 65.6 %
  - 組込デバイスのベース OS （多くの CPU で動作する）
- 「Debian」ベースな派生 OS の源流
  - 例：Ubuntu、Kali Linux、Raspberry Pi OS
  - 派生先のディストリビューションと相互に情報交換をして開発している

- 2024 年 10 月の時点で、最新の安定版は Debian 12.7
  - コードネーム: bookworm
  - パッケージ数は 約 64,419 以上を提供
  - 公式にサポートする CPU アーキテクチャは 9 つ
- コードネームはトイ・ストーリーのキャラクター名を採用
- 次のメジャーリリース は Debian 13
  - コードネーム: trixie
  - 2025 年夏頃にリリースすると思われる

リリースサイクル

- メジャーリリース：おおよそ 2 年ごと
  - Debian は time-based freeze を採用
- 標準サポート：3 年
- LTS[*4]：標準サポート終了後から 2 年
  - 対象アーキテクチャは amd64、i386、arm64, armhf に限定
  - 主要なパッケージのみサポート (全パッケージではない)
- Extended LTS：LTS 終了後から 5 年
  - 公式の Debian プロジェクトの運営ではなく、LTS をメンテナンスしている Freexian の商用サービス
  - 契約者がサポートするパッケージを指定する
  - 更新やセキュリティ修正したパッケージはすべての Debian ユーザーが無料で利用可能
  - お願い：できるだけ次のバージョンに上げてください

2.4 Debian というコミュニティ

Debian への参加者は世界中にいます

- Debian 開発者 (Debian Developer、DD)
  - Debian Project の公式開発者
  - 56 ヶ国に 1,019 名 (2024-10-19 時点)[*5]
  - 日本に在住している人は 33 人 (日本人とは限らない)
  - 国別人数ではドイツ、アメリカ、フランス、イギリス、カナダに次いで第 6 位
- Debian メンテナー (Debian Maintainer、DM)
- パッケージメンテナー
- ドキュメントなどの翻訳

[*3] `https://w3techs.com/technologies/details/os-linux/all/all`
[*4] Long Term Support
[*5] `https://people.debian.org/~eriberto/udd/dd-by-country.html`

- そのほかにも多くの貢献者たちが Debian に参加

- Debian 社会契約[*6]
  - Debian 開発者 たちが目指すフリーソフトウェアコミュニティの在り方
- Debian フリーソフトウェアガイドライン（DFSG）
  - Debian 社会契約の一部
  - Debian が考えるフリーソフトウェアの定義
  - オープンソースの定義のひな形にもなっている
- Debian Policy[*7]
  - Debian パッケージの区分、内容、ルール、ファイル配置の方針などの技術的な定義

2.5 まとめ：Debian とは？

まとめると「Debian」とは

- フリー/オープンな「オペレーティングシステム (OS) 」を作成しようとする人たちが集まるボランティアベースの「プロジェクト」
- 自分たちの考えるフリーという言葉に関する定義、開発目的、パッケージングポリシーを厳格に決めている
- 世界中に 1,000 人以上の公式開発者がおり、他のディストリビューションのベースとして採用されている
- 約 2 年毎にリリースが行われ、多くのパッケージとアーキテクチャをサポートしている
- 上記のような特徴から様々なところで利用されている Linux ディストリビューション

2.6 Debian JP Project と Debian 勉強会

2.6.1 Debian JP Project とは

- `https://www.debian.or.jp/`
- 日本において Debian を普及させることを目的とした任意団体
- 活動内容
  - Debian の日本語による情報発信
  - ユーザとの情報交換
  - Debian 開発者やパッケージメンテナの育成など

2.6.2 Debian 勉強会

- 2005 年 1 月開始で、Debian 開発者 上川さんが発起人
- 月に一回コンスタントに開催している Debian 開発者、Debian ユーザによる勉強会
  - 東京エリア Debian 勉強会
    - `https://tokyodebian-team.pages.debian.net/`
  - 関西 Debian 勉強会
    - `https://wiki.debian.org/KansaiDebianMeeting`
  - COVID-19 の影響もあり、2020 年 3 月からは東京エリアと関西でオンラインによる合同勉強会を開催中
- なぜ始めたか？
  - 当時は「ML と IRC で情報交換」、「face-to-face で会う場所がない」、「まとまったドキュメントが出てこない」の課題あり

[*6] `https://www.debian.org/social_contract`
[*7] `https://www.debian.org/doc/debian-policy/`

  - 勉強会をすることで「定期的に集まる」「資料を作成して公開 (GPL-2+)」[*8]

2.7 Debian 勉強会: 最近の話題

- 勉強会の内容
  - Debian で気になった事柄を調べてレポートする
  - Debian 界隈やパッケージング関連の話題など専門の人に話を聞く、意見を出し合う
- 各地のイベントで Debian 普及活動
  - OSC2024 東京、OSC2024 北海道

2.8 Debian 13 trixie 概要

Debian 13 (コードネーム：trixie)

- `https://wiki.debian.org/DebianTrixie`
- 現在開発を進めている次の安定版リリース
- Debian は time-based freeze を採用しており、およそ 2 年毎のリリースを目指す
  - 2025 年夏頃のリリースと思われる
- サポートするアーキテクチャはまだ確定していない
  - 参考：Debian 12 でサポートしているアーキテクチャ
    - amd64, i386 (i686 以降)
    - arm64, armhf, armel
    - mips64el
    - mipsel （Debian 13 ではリリースなしの予定）
    - ppc64el
    - s390x
  - Debian 13 で新たに追加サポート予定のアーキテクチャ
    - riscv64

2.9 Debian 13 trixie: テーマ

Debian 13 のテーマ

- `https://wiki.debian.org/DebianDesktop/Artwork/Trixie`
- 次期安定版に採用するテーマは公募を行いコンテストを行う (募集の締切は 2024/9/19)[*9]
- 6 件の応募があり、投票の準備中

2.10 Debian 13 trixie 32bit 版変更関連

2.10.1 Debian 13 trixie: 64bit time_t 移行

2038 年問題に対応するべく 32bit 版で仕様変更

[*8] `https://salsa.debian.org/tokyodebian-team/monthly-report`
[*9] `https://lists.debian.org/debian-devel-announce/2024/06/msg00003.html`

- linux kernel、glibc で対応が完了
  - linux-5.6 から 32bit 版/64bit 版ともに UNIX time を 64bit で扱うよう変更
  - glibc-2.34 から 32bit time_t / 64bit time_t 両方の関数を持つよう変更 (32bit 版の既定は 32bit time_t のまま)
- Debian 13 での変更
  - unstable では 32bit 版のパッケージを 64bit time_t を使うよう再ビルドを実施中[*10]
  - 64bit time_t を使うライブラリはパッケージ名の後ろに "t64" がつくように変更 (全 arch で変更)
- ユーザや開発者がやらないといけないこと
  - debian/control などの依存パッケージ名を "t64" を含んだ名前に変える必要がある
  - 64bit time_t に変わっても動くようにテストや改修が必要

2.10.2 Debian 13 trixie: 32bit 版の提供内容変更

32bit 版では提供する機能に削除や制限がある予定
(Debian 12 の 32bit 版は i386、armel、armhf、mipsel を提供)

- アーキテクチャの削除
  - 2023/08/06 testing と unstable で mipsel を削除[*11]
- kernel パッケージの提供取り止め (linux-6.11 以降)
  - 2024/09/14 linux-signed-i386 [*12]
- debian installer の提供取り止め
  - 2024/09/13 armel [*13]、2024/10/08 i386 [*14]
- armhf は (現時点では) 変更なし

2.11 最近の Debain 安定版のリリース情報

- 2024/08/31: Updated Debian 11: 11.11 released
- 2024/08/31: Updated Debian 12: 12.7 released
- 2024/08/14: Security support for Bullseye handed over to the LTS team
- 2024/06/29: Updated Debian 11: 11.10 released
- 2024/06/29: Updated Debian 12: 12.6 released
- 2024/06/15: Debian 10 Long Term Support reaching end-of-life
- 2024/02/10: Updated Debian 11: 11.9 released
- 2024/02/10: Updated Debian 12: 12.5 released

2.12 Bits from the DPL

DPL (Debian Project Leader) からの定期報告

- `https://bits.debian.org/tag/dpl.html`
- 2024 年度の DPL である Andreas Tille さん[*15]がプロジェクトの進捗を報告

[*10] ただし、i386 版だけは適用を除外
[*11] `https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=1043114`
[*12] `https://lists.debian.org/debian-kernel/2024/09/msg00213.html`
[*13] `https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=1081698`
[*14] `https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=1084783`
[*15] 選挙結果 `https://www.debian.org/vote/2024/vote_001`

- 時間のない方でもこれを読んでおけば Debian Project の大まかな動きがわかる
- 最近の投稿
  - 2024/10/07
  - 2024/09/02
  - 2024/08/02
  - 2024/07/20
  - 2024/05/02

2.13 DebConf/DebConf24

DebConf 年に一度、Debian 開発者が集まって開催するカンファレンス

- 2024/07/28 - 08/04: Debconf24: Busan, South Korea
  - 会場：Pukyong National University
  - `https://debconf24.debconf.org/schedule/`
  - 発表のビデオアーカイブがあります
- 2025/07/14 - 07/21 (予定): Debconf25: Brest, France
  - `https://debconf25.debconf.org/`
- 2026 年開催の DebConf は 2024 年の年末年始頃に開催国をコンペする予定

2.14 DebConf24 Busan, South Korea (1)

DebConf24

画像出典：`https://debconf24.debconf.org/`
画像出典：`https://www.debian.org/`
DebConf24 全体スケジュール

- 7/28(日) DebConf Day 1
- 7/29(月) DebConf Day 2 / Cheese and Wine Party
- 7/30(火) DebConf Day 3
- 7/31(水) DebConf Day 4 / Day Trip
- 8/01(木) DebConf Day 5 / Conference Dinner
- 8/02(金) DebConf Day 6
- 8/03(土) DebConf Day 7 / Closing Ceremony
- 8/04(日) (no schedule)
- 8/05(月) Departure day

DebConf24 1 日の流れ

- 起床

- 08:00-10:00 朝食
- 10:00-12:00 午前セッション
- 12:00-14:30 昼食
- 14:30-18:00 午後セッション
- 18:00-20:00 夕食
- 以後、自由行動、就寝

DebConf24 イベント写真

2.15 日本語による Debian の情報

- Debian JP Project
  `https://www.debian.or.jp`
- 東京エリア Debian 勉強会
  `https://tokyodebian-team.pages.debian.net/`
- 関西 Debian 勉強会
  `https://wiki.debian.org/KansaiDebianMeeting`
- Twitter
  `@debian_jp`、`@debianjp`
- 日本語のメーリングリスト
  `https://www.debian.or.jp/community/ml/openml.html`

3 WSL2 での sid 生活

佐々木洋平

3.1 はじめに

... あるいは, なぜこんな事になっているのか, 的な.

昨年の夏に Let's Note CF-SR3 を購入しました。ここ十数年はメイン環境としてレッツノートを利用しており, これまでの機材は Debian を常用する機材として特にハマり所も無かったので, いそいそと Debian をインストールしたわけですが...。

1. SecureBoot できません[*16]。
2. Linux 使用時に idle 時の CPU FAN の回転数が下がりません。

特に二つ目が面倒で, ちょっとそのままでは持ち歩けません。周囲の皆様に騒音でご迷惑をかけそうで, おいそれと開くことができません[*17]。

とはいえ, 私の普段の生活は「ターミナル と Emacs(`emacs -nw`)」です。メールの読み書き・コード書きといった日常業務が遂行できないのは非常に困ります。

20 年前ぐらいに VAIO の某製品を購入して X が起動しなかった時には 1 年ぐらい Cygwin 生活をしたのでまたそれかなぁとも思ったのですが, 「WSL2 では systemd が動くようになったらしいよ[*18]」ということなので, それなら WSL2 を試してみるかと, 周辺知識を調べつつ, いそいそとセットアップを開始しました。

... というわけで, 普段使いが unstable な人がどうにか WSL2 で生活環境を整えるまでの顛末についてお話してみます。Debian 固有の話はあまり出てこないんじゃないか (?) と思いますが, 私の様に WSL2 で Debian を使い倒そうと思っている方の御参考になれば幸いです。対応する WSL の version は以下の通りです。

図 1 `wsl --version` の出力結果

[*16] これまで購入してきた CF-S9, CF-SX3, CF-RZ6, CF-RZ8 などは特にハマらず利用できています。BIOS/UEFI の機能が増えたのか, そもそも Debian Installer だと Boot できないんですよね。そのうち時間をとってジタバタしてみたいですが, 一応メイン機材なので格闘する時間がとれず, 現状は諦め気味です。非常に悔しい。

[*17] Panasonic に問い合わせたら, スペックシートとか公開してくれたりしないかしらん。

[*18] `https://devblogs.microsoft.com/commandline/systemd-support-is-now-available-in-wsl/`

3.2 WSL2

「WSL とはなんですか」というお話は本家 Microsoft の記事[*19] なんかで読めるとは思います。ただ, これらの記事は ~~日本語記事が機械翻訳からの翻案になっていて質が良くないのは置いておくとして,~~ 「こうやったら動きますよ」がメインなので, 我々の様な逸般人 (?) が知りたいであろう内部構造について少しまとめておきましょう。

先ず, WSL には WSL1 と WSL2 の 2 つがあります。現在特に何も考えずに利用すると WSL2 が導入されますが, 用途によっては WSL1 を使いたい場合も未だにあるかもしれません。WSL バージョンの比較[*20]にある機能比較表は以下の通りです。

機能	WSL1	WSL2
Windows と Linux の統合	☑	☑
高速の起動時間	☑	☑
従来の仮想マシンと比較して小さなリソース フット プリント	☑	☑
現在のバージョンの VMware および VirtualBox での実行	☑	×
マネージド VM	×	☑
完全な Linux カーネル	×	☑
システム コールの完全な互換性	×	☑
OS ファイル システム間でのパフォーマンス	☑	×
systemd サポート	×	☑
IPv6 サポート	☑	☑

表 1　WSL1 と WSL2 の機能の比較

この表の項目がそれぞれ何を意味しているのかを説明するために, 以下の図を作成してみました。

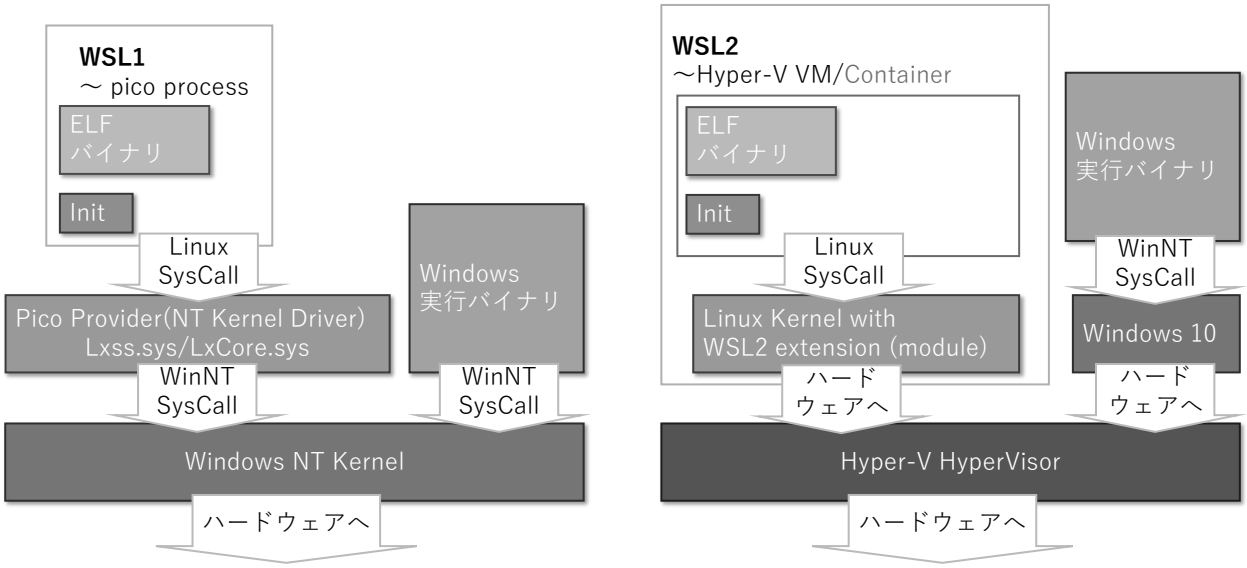


図 2　WSL1 と WSL2 の構造. `https://github.com/ionescu007/lxss` の資料を元に作成

WSL1 は Windows NT kernel 上で ELF バイナリを動作させるための互換レイヤー (Lxss.sys, LxCore.sys) の上に, Linux ディストリビューションのデータを載せて提供しています。図中の init はディストリビューションの提供している init ではなく, WSL1 が提供している init であり, いわゆる deamon の起動はしません。また, WSL1

[*19] 例えば `https://learn.microsoft.com/ja-jp/windows/wsl/` あたり？
[*20] `https://learn.microsoft.com/ja-jp/windows/wsl/compare-versions`

は VolFS という独自のファイルシステム上にデータが置かれます。VolFS ではファイルの実態は NTFS 上に置かれ, Linux でのパーミッション情報などのメタデータは別途管理されています。Windows 側とのデータのやりとりも DrvFS という独自のファイル形式で管理されています。WSL1 からはドライブレター付きでマウント可能であり, WSL2 との違いとして USB メモリ等もマウントできたりします。なお, DrvFS へのファイルの読み書きは Plan9 由来の 9P server を経由して動作します。9P サーバは WSL1 の提供する init に含まれており, WSL1 起動時に起動しています。

WSL2 は Hyper-V の仮想マシン上で Microsoft が手を入れた Linux Kernel[*21]を動作させることで, ほぼ完全な状態の Linux ディストリビューションが動作しています (していることになっています)。設定次第ですが諸々の daemon も動作させることができます。ネットワークは Hypver-V の仮想スイッチが有効になっており, 初期状態では NAT として動作しています[*22]。また, ファイルシステムは WSL1 とは異なり仮想ディスクを利用します。そのため (オーバーヘッドはあるものの)ext4 やそれ以外のファイルシステムも利用可能です。なお, WSL1 とは異なり DrvFS を利用して Windows 側へアクセスするのではなく Windows 側にも 9P サーバが用意されており, 9P を利用して (ネットワークファイルとして) Windows 側のファイルへアクセスすることになります。

以上, ざっくりメリットデメリットまとめると以下の様になるでしょうか?

バージョン	ベースとなる技術	メリット	デメリット
WSL1	システムコールの変換	メモリ消費が少ない	• 命令変換によるオーバーヘッド • 不完全なシステムコール対応 • ディスクアクセスが遅い
WSL2	Hyper-V 仮想化	• システムコール完全互換 • ディスクアクセス速度の改善	メモリ消費量の増大

表 2　WSL1 と WSL2 の機能比較まとめ

構造が違うので, 現時点では WSL1 と WSL2 は共存できる様です。人によっては用途によって使い分けていたりする様です。

... とはいえ, 自分の用途では

- メールを読むのに mbsync で定期的に手元にメールを取得
- 手元で Dovecot を上げておき, Emacs 上の Wanderlust からメールを読む
- メールの送信は exim4 の smarthost に丸投げ

などといった, daemon を常駐させる様な使い方がほとんどです。そんなわけで WSL2 を導入して設定を開始しました。「WSL2 では systemd が動くようになったらしいよ[*23]」ということなので, きっと大丈夫だろう, という浅はかな考えから始めたのですが...

3.3　ここが変だよ? WSL2

3.3.1　systemd までの道程

導入は Windows の Powershell にて

```
PS C:\Users\uwabami> wsl --install -d Debian
```

[*21] https://github.com/microsoft/WSL2-Linux-Kernel
[*22] 他にもブリッジモードやミラーモードなどがありますが, 上手く使えておりません。
[*23] 再掲: https://devblogs.microsoft.com/commandline/systemd-support-is-now-available-in-wsl/

で OK です。初回起動時にユーザ名とパスワードが聞かれ, ログインできる様になります。

この時点では systemd が動作していませんので, 先ずは `/etc/wsl.conf` を作成し systemd を起動することにします。

```
% vi /etc/wsl.conf
...
% cat /etc/wsl.conf
[boot]
systemd=true
```

その後 WSL を再起動しましょう。

```
$ systemctl status
Failed to connect to bus: No such file or directory
```

アッハイ。

```
$ sudo apt update
$ sudo apt install dbus-user-session
```

ということで, 気をとりなおして WSL を再起動して...

```
$ systemctl status
Failed to dump process list for 'azalea', ignoring: Input/output error
azalea
    State: running
    Units: 280 loaded (incl. loaded aliases)
     Jobs: 0 queued
   Failed: 0 units
    Since: Sat 2024-08-17 12:28:10 JST; 8s ago
  systemd: 252.26-1~deb12u2
  Tainted: cgroupsv1
   CGroup: /
```

ふむ。**Tainted: cgroupsv1** ということで, `/etc/fstab` に以下を追加しておきます。

```
$ vi /etc/fstab
...
$ cat /etc/fstab
cgroup2 /sys/fs/cgroup  cgroup2 rw,nosuid,nodev,noexec,relatime,nsdelegate      0       0
```

その後で再起動して試すと...

```
$ systemctl status
Failed to dump process list for 'azalea', ignoring: Input/output error
azalea
    State: running
    Units: 280 loaded (incl. loaded aliases)
     Jobs: 0 queued
   Failed: 0 units
    Since: Sat 2024-08-17 12:31:44 JST; 22s ago
  systemd: 252.26-1~deb12u2
   CGroup: /
```

... はて。

ということでちょっと調べてみると

- systemd status fails output on Ubuntu #8879 `https://github.com/microsoft/WSL/issues/8879`

にある通りで, 結論から言えば `systemd 255.7` 以降なら大丈夫です[*24]。`/etc/apt/sources.list` を書き換えて, いそいそと sid にアップグレードしましょう。

```
$ sudo vi /etc/apt/sources.list
...
$ cat /etc/apt/sources.list
deb http://deb.debian.org/debian sid main
$ sudo apt update
$ sudo apt-get upgrade
$ sudo apt-get dist-upgrade
```

[*24] WSL のデフォルトが Ubuntu 22.04 のままなのは, こういう事情から, かと思います。

というわけで, 一通り導入が終わったら...

```
  $ cat /etc/debian_version
  trixie/sid
  $ systemctl status
azalea
   State: running
   Units: 308 loaded (incl. loaded aliases)
    Jobs: 0 queued
  Failed: 0 units
   Since: Sat 2024-08-17 12:44:08 JST; 23s ago
 systemd: 256.5-1
 Tainted: unmerged-bin
  CGroup: /
          ├─ init.scope
          │  ├─  1 /sbin/init
          │  ├─  2 /init
          │  ├─  7 plan9 --control-socket 6 --log-level 4 --server-fd 7 --pipe-fd 9 --log-truncate
          │  ├─ 198 /init
          │  ├─ 199 /init
          │  ├─ 200 -bash
          │  ├─ 223 systemctl status
          │  └─ 224 less
          ├─ system.slice
          │  ├─ console-getty.service
          │  │  └─ 185 /sbin/agetty -o "-p -- \\u" --noclear --keep-baud - 115200,38400,9600 vt220
          │  ├─ cron.service
          │  │  └─ 162 /usr/sbin/cron -f
          │  ├─ dbus.service
          │  │  └─ 166 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation --syslog-o
          │  ├─ system-getty.slice
          │  │  └─ getty@tty1.service
          │  │     └─ 186 /sbin/agetty -o "-p -- \\u" --noclear - linux
          │  ├─ systemd-journald.service
          │  │  └─ 38 /usr/lib/systemd/systemd-journald
          │  ├─ systemd-logind.service
          │  │  └─ 171 /usr/lib/systemd/systemd-logind
          │  └─ systemd-udevd.service
          │     └─ udev
```

ようやくスタートラインです。

3.3.2 時計合わせ

少し使ってみて気がついたのですが, 起動時はともかく, Host の Windows 側の susupend & resume 後に時刻がズレます。仮想環境あるあるですね。

WSL 側で時刻合わせをしても良いですけれど, Host 側と時刻合わせのタイミングがズレるのはちょっと嫌なので, どうしたモンかな, とすこし悩みました。結論から言えば, WSL2 環境では `/dev/ptp0` が生えていますので, これを使えば良い, ということになります。

- Time sync for Linux VMs in Azure:
  `https://github.com/MicrosoftDocs/azure-docs/blob/main/articles/virtual-machines/linux/time-sync.md`

`systemd-timesyncd` にはまだ `/dev/ptp0` のサポートは無いようです。

- PTP support for timesyncd #22828 `https://github.com/systemd/systemd/issues/22828`

というわけで, 安直には chrony を利用します。上記 Azure のドキュメントにある通り, 設定ファイルに以下を追記して chrony を起動させておきます。

```
% vi /etc/chrony.conf
...
# Step the system clock instead of slewing it if the adjustment is larger than
# one second, but only in the first three clock updates.
# makestep 1 3          # <- コメントアウト
makestep 1.0 -1         # <- 追記
...

## use ptp0             # 以下, 追記
refclock PHC /dev/ptp_hyperv poll 3 dpoll -2 offset 0 stratum 2
```

これで, Windows 側との時刻同期が行なわれる様になる (筈) です。あとは Windows 側で適宜 NTP サーバ等の設定をしておきます。

3.3.3 keyring をどうするか?

ssh-agent, gpg-agent, secret-tools などの, いわゆる「鍵束」をどう使うかに悩みました。Linux の Desktop 環境の場合はログイン認証時に PAM を経由して gnome-keyring の鍵束が unlock されます。これを利用するのが良くある話かとは思いますが, そもそも WSL にはログイン認証がありません。

結局,

- ssh-agent と gpg-agent については `keychain` パッケージを使う
- secret-tools については WSLg があるので gnome-keyring を使う。
  - ログイン時にシェルから呼び出して一度 unlock しておく

といった手段を取る事にしました。

なお **ログインシェルからの呼び出し時の pinentry では何故か GUI のキー配列が US 配列になります**。常に US 配列となるわけではなく, 2 回目・3 回目の pinentry の呼び出し時には (言語か何かを見たのか) 日本語配列となります。... なんでですかねぇ。

3.3.4 快適なターミナルを求めて

快適にターミナルで生活するための条件は人によって異なるとは思いますが, 私の場合は

- East Asian Ambiguous Width が制御できること。
- Emoji がちゃんと表示できること。特に Zero Width Joiner をちゃんと処理できること。

が上げられます。前者はターミナルと Emacs の文字幅を揃えるために, 後者は Weechat などで崩れずに表示するために, でしょうか[*25]。現時点では

- 表示が一番良いのは WSLtty (mintty の WSL 対応版)
- 表示がマシで, 処理速度が軽快なのは WezTerm

ということで, 基本的に WezTerm を利用しています。とはいえ WezTerm の公式ドキュメントにある通り[*26], WezTerm をそのまま起動して wsl を叩くと, 折角の文字処理が全て ConPTY に持っていかれて悲しい目に合いますので, WSL2 を起動したら, Wezterm 内蔵の SSH で接続して利用しています。sshd の設定で `AcceptEnv` を

```
$ cat /etc/sshd_config
...
# Allow client to pass locale environment variables
# AcceptEnv LANG LC_*
AcceptEnv LANG LC_* TERM COLORTERM TERM_PROGRAM TERM_PROGRAM_VERSION WSL* WEZTERM*
...
```

と幾つか追加しておくと良いでしょう。

3.4 おわりに

というわけで, 生活環境としての WSL2 の設定でハマった所について簡単にまとめてみました。動作させ始めてからは, まあ普通の環境かな, 学生に実習などで利用してもらう上では非常に便利かな, とか思いつつも, ネットワーク周りがまだまだちゃんと整理できておらず, ちょっとモヤモヤしていたりします (特に VPN 利用時のネットワーク周りの設定が良くわかっていません)。

WSL2 はこれからまだまだ更新されていくとは思いますので, 現時点での情報として御笑覧頂ければ, 幸いです[*27]。

[*25] ついでに, 動作が軽快であるとか, 余計な処理をしないとか, 余計な装飾をしない, とか。イロイロ求め出すとキリが無いんですが, まあそれはそれとして。

[*26] Windows and ConPTY: `https://wezfurlong.org/wezterm/what-is-a-terminal.html?h=conpty#windows-and-conpty`

[*27] なお, 別に WSL2 が好きな訳じゃないくて, ちゃんと動作させられるなら素の Debian 環境に移行したいな, とは思ってます。次に買うならなんだろう。やっぱり ThinkPad かな。X1 Nano Gen3 って Linux 動くんだろうか?

4 autopkgtest について調べてみた

Norimitsu SUGIMOTO (杉本 典充)

4.1 目次

アジェンダ

- 自己紹介
- おさらい：debian パッケージのビルド方法
- autopkgtest とは
- autopkgtest を用いたテストケースの作成と実行
- 仮想環境における autopkgtest の実行方法
- まとめ
- 参考資料

4.2 自己紹介

- Norimitsu SUGIMOTO (杉本 典充)
- dictoss@live.jp
- Twitter: @dictoss
- Debian を使い始めたのは 3.1 sarge が testing の頃
- 仕事はソフトウェア開発者をやってます
- python と Django の組み合わせで使うことが多いです

4.3 おさらい：debian パッケージのビルド方法

4.3.1 debian ディレクトリの中身

Debian 新メンテナーガイド[*28] には.deb ファイルの元になる src パッケージの中の debian ディレクトリについて説明がある

- debian/control
- debian/copyright
- debian/changelog
- debian/rules

[*28] https://www.debian.org/doc/manuals/maint-guide/

- debian/watch
- debian/source/format

Debian 新メンテナーガイドには書いていませんが、以下のファイルも扱える

- debian/upstream/metadata
- **debian/tests/control** ←今回はこのファイルの話

4.3.2 ビルド方法

debian パッケージのビルドに必要なツールをインストール

```
$ sudo apt-get install build-essential devscripts
```

既存の grep パッケージを自分でビルドしてみる場合

```
$ sudo apt-get build-dep grep
$ apt-get source grep
$ cd grep-3.11
$ debuild -uc -us
$ ls ../grep*.deb
../grep-dbgsym_3.11-4_amd64.deb  ../grep_3.11-4_amd64.deb
```

ビルドした debian のバイナリパッケージの中身がきちんと動くかを確認するには？
→ x 自分の環境にインストールして頑張ってテストする
→ o debian/tests/control を作成し、autopkgtest でテストする

4.4 autopkgtest とは

- インストールされたバイナリパッケージをソースパッケージに付属するテストケースを用いてテストする方法
- 「# apt-get install autopkgtest」でインストールできる
- 詳しい説明
  - man autopkgtest
  - `https://wiki.debian.org/ContinuousIntegration/autopkgtest`
  - `https://people.debian.org/~eriberto/README.package-tests.html`
  - `https://ci.debian.net/doc/file.TUTORIAL.html`
- autopkgtest があると unstable から testing への移行が早い、unblock リクエストが不要な場合あり
  - `https://release.debian.org/bookworm/freeze_policy.html`

4.4.1 autopkgtest とは

- テストベッド：テスト対象のバイナリパッケージをインストールしたテスト環境
  - ローカル環境を使う方法
  - 仮想環境を使う方法 (schroot / lxc / qemu)
- autopkgtest のベストプラクティス[*29]
  - 【重要】テストケースは src パッケージにあるが、テスト対象は src パッケージ内のビルド成果物ではない
  - テストは実行されるシステム (=テストベッド) を変更する可能性がある
  - upstream のテストが付属する場合はそれを使うことを推奨
  - 各プログラム言語で利用可能な標準的なテストソリューションを使用する (shunit2, nose, prove)
  - 1 件当たりのテストは短い時間で完了させるようにする (タイムアウトが起こります)

[*29] `https://wiki.debian.org/ContinuousIntegration/AutopkgtestBestPractices`

4.5 autopkgtest を用いたテストケースの作成と実行

4.5.1 autopkgtest の定義の例

grep パッケージにおける src パッケージの debian/tests

```
$ ls -ln debian/tests/
合計 20
-rw-rw-r-- 1 1000 1000 309 11 月 23  2023 control
-rw-rw-r-- 1 1000 1000 234 11 月 23  2023 dirfd
-rw-rw-r-- 1 1000 1000  93  8 月 23  2023 egrep-fgrep
-rwxrwxr-x 1 1000 1000 318  8 月 23  2023 stray-backlash-warn
-rw-rw-r-- 1 1000 1000 240  1 月  4 20:56 upstream-test-suite
```

debian/tests/control の Tests キーの値の名前がテストケースで、debian/tests/ 配下のスクリプトファイル名に対応

```
$ cat debian/tests/control
Tests: upstream-test-suite
Depends: @builddeps@, fakeroot, libpcre2-8-0, locales-all
# TODO: remove allow-stderr. For now it is needed since for being able
#       to ignore expensive checks and for ignore unavailable locale
#       warnings.
Restrictions: allow-stderr

Tests: egrep-fgrep, stray-backlash-warn dirfd
```

4.5.2 autopkgtest の実行方法

- パッケージをビルド、インストールし、autopkgtest を実行する
- autopkgtest を実行して終了コードが 0 であればテストケースはすべて合格
- オプションの ”null” は テストベッド環境を指定するオプションでローカル環境を差す

```
$ cd grep-3.11
$ debuild -uc -us
$ sudo dpkg -i ../grep_3.11-4_amd64.deb
$ autopkgtest --no-built-binaries -- null
  （テスト処理の出力を表示）
$ echo $?
0
```

4.5.3 autopkgtest のテストケースの作成 (1)

テストケースを作成する

- 実行するコマンドの終了コードが成功 (0) になるテストコードを書く
- set -e を指定するなど、途中のテストで失敗した場合は異常終了させる

```
$ vi debian/tests/mytest1
#!/bin/sh
set -e

# test egrep and -E option
printf 'aabbccdd' | grep -E 'a{2,}'
printf 'aabbccdd' | egrep 'a{2,}'
```

debian/tests/control に追加したテストを加える

```
Tests: egrep-fgrep, stray-backlash-warn dirfd, mytest1
```

4.5.4 autopkgtest のテストケースの作成 (2)

作成したテストケースを単体で実行 (テストに合格した場合)

```
$ autopkgtest --no-built-binaries --test-name=mytest1 -- null
autopkgtest [23:42:00]: starting date and time: 2024-06-20 23:42:00+0900
autopkgtest [23:42:00]: version 5.35
autopkgtest [23:42:00]: host nor-deb-sid-1; command line:
  /usr/bin/autopkgtest --no-built-binaries --test-name=mytest1 -- null
autopkgtest [23:42:00]: testbed dpkg architecture: amd64
autopkgtest [23:42:00]: testbed apt version: 2.9.5
autopkgtest [23:42:00]: testbed running kernel: Linux 6.8.12-amd64 #1
  SMP PREEMPT_DYNAMIC Debian 6.8.12-1 (2024-05-31)
autopkgtest [23:42:00]: @@@@@@@@@@@@@@@@@@@@@@ built-tree .
autopkgtest [23:42:00]: testing package grep version 3.11-4
autopkgtest [23:42:00]: test mytest1: preparing testbed
autopkgtest [23:42:00]: test mytest1: [-----------------------
aabbccdd
aabbccdd
autopkgtest [23:42:00]: test mytest1: -----------------------]
autopkgtest [23:42:00]: test mytest1:  - - - - - - - - - - results - - - - - - - - - - -
mytest1              PASS
autopkgtest [23:42:00]: @@@@@@@@@@@@@@@@@@@@@@ summary
mytest1              PASS

$ echo $?
0
```

4.5.5 autopkgtest のテストケースの作成 (3)

作成したテストを単体で実行 (1 行目のテストが失敗した場合)

```
$ vi debian/tests/mytest1
#!/bin/sh
set -e

# test egrep and -E option
printf 'aabbccdd' | grep -E 'e{2,}'
printf 'aabbccdd' | egrep 'a{2,}'
```

```
$ autopkgtest --no-built-binaries --test-name=mytest1 -- null
  (省略)
autopkgtest [22:52:17]: test mytest1: [-----------------------
autopkgtest [22:52:18]: test mytest1: -----------------------]
autopkgtest [22:52:18]: test mytest1:  - - - - - - - - - - results - - - - - - - - - - -
mytest1              FAIL non-zero exit status 1
autopkgtest [22:52:18]: @@@@@@@@@@@@@@@@@@@@@@ summary
mytest1              FAIL non-zero exit status 1

$ echo $?
4
```

4.6 仮想環境における autopkgtest の実行方法

4.6.1 autopkgtest のテストベッド環境

- ローカル環境でテストを実行する場合の問題点
  - ローカル環境で実行する場合は　ビルド環境 ＝ テストベッド環境　になる
  - バイナリパッケージをインストールしてテストする性質上、テストするパッケージが壊れていた場合は自分のビルド環境を壊す可能性あり
  - 重要なコマンドやライブラリを含むパッケージはテストしにくい
- ビルド環境とテストベッド環境を分離するために仮想環境を利用できる
  - schroot (準備するのが手軽)
  - lxc (`https://ci.debian.net/` で採用)
  - qemu (仮想サーバの上では仮想環境がネストになり実行できない可能性あり)

今回は schroot 環境で実行する手順を紹介します

4.6.2 schroot 環境の準備 (1)

コンテナ環境を debootstrap を使ってインストール

```
$ sudo apt-get install debootstrap schroot
$ sudo mkdir /srv/chroot
$ sudo debootstrap unstable /srv/chroot/sid1 http://deb.debian.org/debian/
$ ls /srv/chroot/sid1
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt
proc  root  run  sbin  srv  sys  tmp  usr  var
```

/etc/schroot/chroot.d 配下の設定ファイルを作成 ([セクション] の中がコンテナ名)

```
$ sudo vi /etc/schroot/chroot.d/sid1.conf
[my-sid1-amd64]
description=Debian unstable amd64
directory=/srv/chroot/sid1
users=dictoss
root-groups=dictoss
aliases=unstable,default
type=directory
profile=desktop
union-type=overlay
```

4.6.3 schroot 環境の準備 (2)

schroot で利用できる環境一覧に表示されます

```
$ schroot -l
chroot:default
chroot:default-source
chroot:my-sid1-amd64
chroot:unstable
chroot:unstable-source
source:default
source:my-sid1-amd64
source:unstable
```

autopkgtest コマンドで 前は ”null” だった部分を ”schroot” に変更

```
$ cd grep-3.11
$ debuild -uc -us
$ autopkgtest --no-built-binaries -- schroot my-sid1-amd64
-> テストケースの実行に必要なパッケージのインストール、ビルド、テストが始まる
...
```

4.6.4 schroot 環境で autopkgtest

autopkgtest の実行出力

```
...
autopkgtest [00:15:37]: test mytest1: preparing testbed
Reading package lists... Done
Building dependency tree... Done
Starting pkgProblemResolver with broken count: 0
Starting 2 pkgProblemResolver with broken count: 0
Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
autopkgtest [00:15:37]: test mytest1: [-----------------------
aabbccdd
aabbccdd
autopkgtest [00:15:38]: test mytest1: -----------------------]
autopkgtest [00:15:38]: test mytest1:  - - - - - - - - - - results - - - - - - - - - - -
mytest1              PASS
autopkgtest [00:15:38]: @@@@@@@@@@@@@@@@@@@@@ summary
upstream-test-suite  PASS
egrep-fgrep          PASS
stray-backlash-warn  PASS
dirfd                PASS
mytest1              PASS
```

4.7 まとめ

- autopkgtest のインストール方法、実行方法、テストの書き方を説明しました
- autopkgtest を仮想環境で実行する 1 つの例 (schroot) の方法を紹介しました
- 来年 2025 年は Debian 13 trixie のリリースに向けてフリーズが始まる見込みで、autopkgtest を書いておくと更新したパッケージの testing 移行が早い
- autopkgtest を書いて、品質のよいパッケージを作ってみましょう

4.8 参考資料

- Debian Wiki - ContinuousIntegration autopkgtest
  https://wiki.debian.org/ContinuousIntegration/autopkgtest
- man autopkgtest-virt-schroot(1)
  https://manpages.debian.org/testing/autopkgtest/autopkgtest-virt-schroot.1.en.html
- Charles Plessy. Debian Continuous Integration, 第 152 回東京エリア Debian 勉強会, 2017/06
  https://tokyodebian-team.pages.debian.net/pdf2017/debianmeetingresume201706.pdf
- Youhei SASAKI. 「パッケージ作成環境: sbuild, lintian, piuparts, autopkgtest」
  https://uwabami.github.io/cc-env/DebianPackaging.html

5 General Resolution:tag2upload は Debian にどんな変化をもたらすのか

Kentaro Hayashi(林健太郎)

本記事は Debian 勉強会 (2024 年 7 月開催) にて公開した資料を冊子向けに編纂・収録したものです。
なお、2024 年 12 月時点では、tag2upload のしくみはまだ正式に導入されていません。今後の進展が待たれます。

5.1 General Resolution: tag2upload とは

2024/06/27 に Sean Whitton 氏によって tag2upload の扱いをどうするかが Debian General Resolution として提案が行われました。

Debian General Resolution (以下 GR) とは Debian での方針を Debian 開発者による投票によって決定する制度です。毎年実施している Debian プロジェクトリーダーの選挙を除くと、通常投票で物事を決定する機会はあまりありません。この GR という制度が利用された例としては、公式インストーラーに自由でないファームウェアを同梱できるようにするかどうかが争われた事例があります。

GR が発動されたということは、Sean Whitton 氏や Ian Jackson 氏が中心になって行われた tag2upload という新しい仕組みについて、Debian で正式採用することを推し進めるための強力なカードを切ったといえるでしょう。

詳細は以下のメールを参照していただくとよいのですが、本記事では概略について説明します。

- GR: tag2upload, text of proposal [*30]
- General Resolution: tag2upload [*31]

[*30] https://lists.debian.org/debian-vote/2024/06/msg00561.html
[*31] https://www.debian.org/vote/2024/vote_002

5.2 提案された tag2upload の内容抜粋

tag2upload allows DDs and DMs to upload simply by using the git-debpush(1) script to push a signed git tag.
1.tag2upload, in the form designed and implemented by Sean Whitton and Ian Jackson, and design reviewed by Jonathan McDowell and Russ Allbery, should be deployed to official Debian infrastructure.
2.Under Constitution §4.1(3), we overrule the ftpmaster delegate's decision: the Debian Archive should be configured to accept and trust uploads from the tag2upload service.
3.Future changes to tag2upload should follow normal Debian processes.
4.Nothing in this resolution should be taken as requiring maintainers to use any particular git or salsa workflows.

本提案の肝要なところは、Debian 開発者 (以下 DD) や Debian メンテナー (以下 DM) が git-debpush(1) を使って git tag に署名するだけで、パッケージをアップロードできるようにする仕組みを実現するところにあります。そのために、次のような段階を踏んで既存のやりかたに変更を加えようとしていました。

- 1.tag2upload を Debian 公式なインフラへと組み込む
- 2.Debian 憲章にもとづいて、tag2upload サービスからのアップロードを信頼できるものとして受け入れるようにする (ftpmaster はこの決定に従ってね)
- 3.tag2upload の将来的な変更は GR とかじゃなくて通常の Debian の開発プロセスに則って実施するよ
- 4. この決定はメンテナに特定の git でのワークフローを強制するものではないよ

tag2upload は git tag に署名されていたらそれをもとにソースパッケージのアップロードまで代行してくれるしくみを導入するものです。(従来は各メンテナーがソースパッケージをビルドしてからアップロードするようになっていました)

GR によって tag2upload の仕組みを ftpmaster に採用してもらうことをねらっていました。なぜ ftpmaster がここでででてくるかというと Debian 公式アーカイブに関する責任を負っているのが ftpmaster であるからです。各開発者が直接アップロードしたのではなく、サービスによる代理アップロードという仕組みになるため、セキュリティを担保する観点からこの仕組みは ftpmaster によって受け入れられないとされてきたという経緯があります。

過去にも ftpmaster と調整を試みてきていたようですが、不調に終わったため GR による解決を図る事態となりました。ftpmaster の方針をひっくり返すには (説得に失敗したら)、現状 GR しかないとされているためです。

5.3 この tag2upload の GR はどうなったか？

メーリングリストで議論を重ねた結果、Re: General Resolution to deploy tag2upload[*32] で説明されているように GR は Sean 氏によって撤回されました。これはもともと提案されていた tag2upload の仕様を変更することにより ftpmaster と合意形成ができたためです。

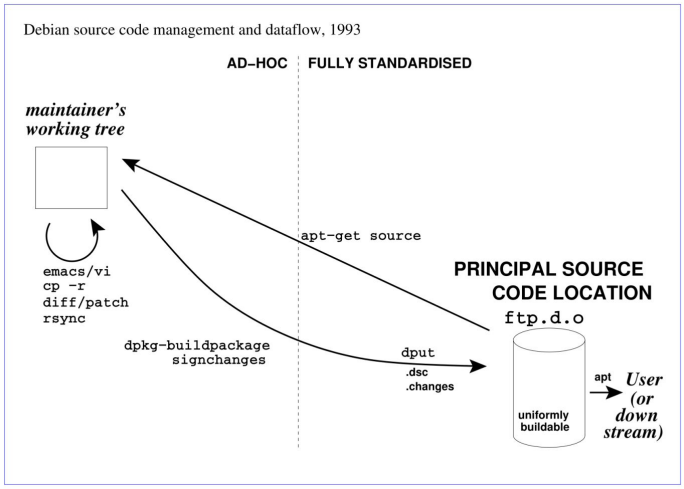
5.4 tag2upload はどういうものなのか

tag2upload についてより詳しく知るには、MiniDebConf Cambridge 2023 (2023 年 11 月開催) のプレゼンが参考になります。

tag2upload - uploads, made automatically and fully traceably, from git [*33]

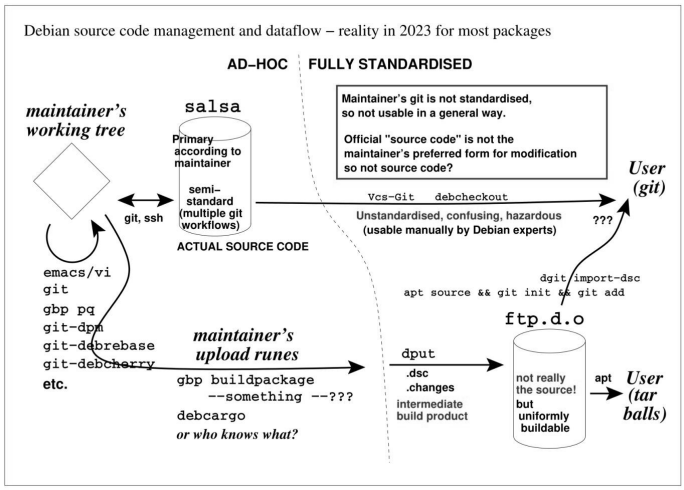
次の図 (動画より引用) は伝統的なパッケージの更新のフローです。

[*32] https://lists.debian.org/debian-vote/2024/07/msg00025.html
[*33] https://www.youtube.com/watch?v=1ohpZ4Xj0iQ



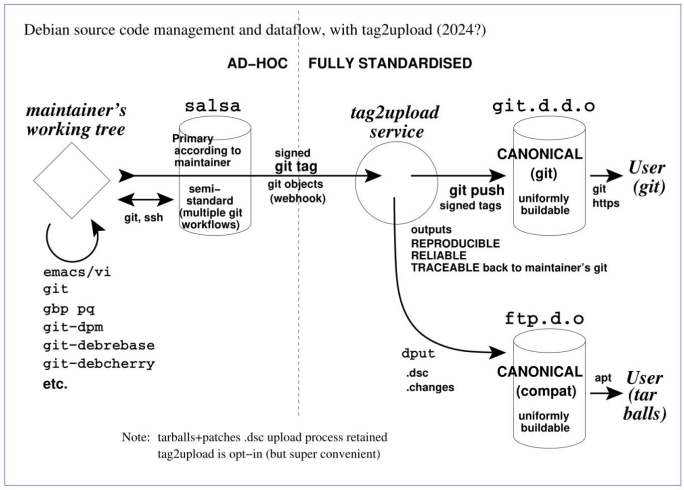
あくまで公式アーカイブから入手できるソースコードが正式なもので、とくにリポジトリによる管理がなされていなかった時代のワークフローです。

最近のパッケージの更新のフローでは次の図 (動画より引用) のようになっています。



多くのパッケージが salsa.debian.org という GitLab のインスタンスにメンテナンスのためのリポジトリを置いています。なんらかの手段でリポジトリの内容からソースパッケージを生成し、成果物を Debian 公式アーカイブへとアップロードするというワークフローです。

tag2upload の導入でどうなるかというと次の図 (動画より引用) のようになります。



パッケージのメンテナンスをするひとは、salsa 上で tag に署名して push するところまでを実施します。ソース

パッケージを生成して、Debian 更新アーカイブにアップロードするところは tag2upload というサービスが実施します。パッケージの開発者は (デバッグや検証目的を除いて)、ローカルの PC 等でアップロードのためにパッケージをビルドする必要すらありません。

tag2upload の進行状況としては以下の作業が完了しています。(2024 月 7 月時点)

- tag2upload のしくみのデザイン
- git tag に署名する git-debpush の実装
- dgit サーバーの稼働 (git.dgit.debian.org)
- ソースパッケージ (.dsc) を生成するしくみの実装
- tag2upload のしくみのレビュー

したがって、tag2upload の残課題は以下のようなものがあります。

- webhook とジョブを動かすサービスの検討 (DSA と議論が必要)
- tag2upload を試験稼働させる (DSA と議論が必要)
- ベータ稼働、ユーザーからのフィードバック等

いずれも、Debian のシステムを管理している DSA との調整が必要な作業が残っています。

5.5 tag2upload が議論となった背景

そもそものこの tag2upload という仕組みを実現するアイデア自体は急にでてきたわけではありません。

2019 年 7 月: Ian Jackson 氏により設計の草稿が示されましたが[*34]、当時 ftpmaster を説得しきれず立ち消えになったという経緯があります。

その後 2024 年 6 月 Sean Whitton さんにより GR の草稿が [RFC] General Resolution to deploy tag2upload[*35] として示されますが、次のような反対意見がなされます。

- tag に署名をするだけでは不十分ではないか (ftpmaster)
- リリースを意図したアーカイブになっていることを担保すべき (例: 手元でビルドしないと得られないメタデータなりチェックサムを付与すべき)

しかし、反対意見について tag2upload 推進派は、tag2upload のコンセプトと相容れないとして一蹴していました。結局 2019 年のときと同じような議論が繰り返されたとして、現状のデザインをそのまま採用させたいことから GR が提案されるにいたりました。

5.6 どのような合意形成が行われたか

t2u in the archive (was: General Resolution to deploy tag2upload)[*36]において、次のような改善をすることで合意にいたったようです。

- タグに署名をするのに加えて tag にひもづいた内容を含める
  (例: `git ls-files --format="%(objectmode) %(objectname) %(path)"`)
- git のアーカイブ (SOURCE_VERSION.git.tar.xz) を.changes に含める
- メンテナーが意図したアーカイブの内容であることを検証できるようにする

あくまであとからパッケージのメンテナーがリリースしようとしたアーカイブの内容であることをちゃんと検証できる状態を担保するというのが重要だったようで、落とし所を探って解決が図られたのはよいことです。

[*34] https://lists.debian.org/debian-dak/2019/07/msg00025.html
[*35] https://lists.debian.org/debian-vote/2024/06/msg00000.html
[*36] https://lists.debian.org/debian-vote/2024/06/msg00602.html

5.7 tag2upload の導入によって何が変わるのか

では、Debian パッケージの開発者にとって tag2upload が導入されることによって何が変わるのでしょうか。
Debian 開発者の視点だと次のようになります。

- ソースパッケージのアップロードは不要になる
- tag を打つときに署名を付与して salsa.d.o に git debpush するだけでよい
- tag2upload は利用しなくてもよい (オプションだから)

tag2upload が導入されても、その利用は強制ではありません。既存のワークフローを変更することになるため、反発も大きいからでしょう。実際、別のトピックとして、Git を利用したパッケージング方法の標準化の話もメーリングリストで議論されてはいますが、そちらもまだまとまりそうにはありません。
一方、ユーザーにとって何が変わるのでしょうか。

- git.dgit.d.o からタグの打たれたソースコードを入手できる
- git を経由してソースを取得し、より再現性のあるビルドが行える (salsa.d.o との比較)

上記のような違いがありますが、一般ユーザーの観点だとあまり影響はないといえるでしょう。
では、Debian プロジェクトにとってどのような意義があるのでしょうか。これは主にトレーサビリティの向上が挙げられます。
ソースパッケージを一貫したやりかたで生成できることが担保されるようになりますし、変なコードの混入を防ぎやすくなります。git.dgit.d.o は追記のみ可能な仕様で運用されることになるはずなので、問題のあるコードが仮に混入したとしてもあとから検証できるはずです。
tag2upload が積極的に採用されたときのメリットとしては、salsa.d.o が現状を正しく反映するようになるというのがあります。salsa.d.o に修正が反映されず、直接ソースパッケージのみが修正されるというのが繰り返されると、salsa.d.o の内容と Debian の公式アーカイブから入手できるソースパッケージに乖離が発生します。
tag2upload は salsa.d.o を経由する設計のため、リポジトリの内容と提供されているパッケージの内容が違うというのを回避しやすくなる（かもしれない）という意義があります。[*37]

5.8 tag2upload を使うには

まだ正式に稼働はしていないものの、こんなふうに使えるというのは案内されています。すでに git buildpackage によるパッケージのメンテナンスのワークフローに慣れている人にとっては簡単に扱えるようになっているようです。

```
$ sudo apt install git-debpush
$ cd (salsa.d.o から clone したディレクトリ)
$ git debpush --gbp
```

5.9 さいごに

tag2upload が採用されることで、パッケージの透明性が高まるかもしれないというのはよい改善点です。しかし、DSA との調整次第なのでいつ導入されるかは不明です。また、tag2upload を採用するかどうかはパッケージのメンテナー次第なところもあるので、どれくらい普及するかは不透明です。リポジトリの運用ポリシーによってはサポートされていないという制約も今後の改善が望まれます。いろいろ課題はありますが、今後に期待したい取り組みのひとつと言えるのではないでしょうか。

[*37] 修正したひとが tag2upload の仕組みを一切利用せず直接ソースパッケージを更新したら意味がないのですが。

5.10 参考

tag2upload に関するあれこれをまとめている記事については Debian debate over tag2upload reaches compromise[*38]を参照するとよいでしょう。とてもよくまとまっています。

[*38] https://lwn.net/Articles/978324/

6 より新しい Mozc を Debian でも使いたい

Kentaro Hayashi(林健太郎)

本記事は Debian 勉強会 (2024 年 11 月開催) にて公開した資料を冊子向けに編纂・収録したものです。
なお、2024 年 12 月時点では、Debian unstable の Mozc は delayed アップロードされているのでまだ 2.28.4715.102 から 2.29.5160.102 に更新されていません。

6.1 Debian における日本語入力

Debian において日本語入力は、インプットフレームワークの違いこそあれほぼ Mozc に支えられているといってもよいでしょう。実際 GNOME では ibus-mozc が優先されますし、その他のデスクトップ環境でも uim-mozc が選択されます。

2010 年に公開された Mozc が今でも継続的にメンテナンスされていることはとても素晴らしいことです。かつては PR の受け入れ範囲の制限があるなど、ライセンスは自由であっても開発スタイルがオープンでないといわれていたこともありました。最近では PR の受け入れ範囲の制限の撤廃がなされるなど、より開発の間口を広げる動きが見られます。[*39]

またビルドシステムの刷新や、UI のフレームワークを GTK から Qt に変更するなどのそれなりに大きな変更も投入されるなどしています。

6.2 Debian における Mozc のメンテナンス状況

2022 年以降はメンテナンスがやや停滞ぎみで、2024 年は FTBFS への対処のみでした。testing から Mozc が削除されてしまっていて、新規にインストールできなくなってしまったこともありました。Mozc がビルドできなくなると、独自にシステム辞書を差し替えて使うためにリビルドするということもできなくなります。

Debian では長らく 2.28.4715.102 というバージョンが採用されていますが、その間 upstream では 2.28 系だけでなく、2.29 系、そして 2.30 系もリリースされています。

upstream への追従は積極的には行われていない状態といえます。これはメンテナンスがボランティアベースであるためなかなか難しい問題です。

他のディストリビューションではどうかというと、Fedora では 2.29.5111.102 が、openSUSE では 2.29.5268.102 を採用していました。(2024 年 11 月時点)

[*39] https://zenn.dev/komatsuh/articles/komatsuh_mozc_updates_from_2023_10

6.3 Debian における Mozc を更新する壁

Mozc を更新するための壁となる要素はいくつかあります。

- Mozc の upstream はビルドシステムを GYP から Bazel に置き換えた
- Debian では Bazel は古いバージョンしかパッケージングされていない
- Mozc の upstream は GYP を 2.29.5544.102 で非推奨としすでに削除をすすめている
- Mozc が protobuf や abseil に依存しているが、新しいバージョンが要求される
- uim や fcitx5 向けのパッチを更新していく必要がある

upstream のバージョンに追従していくには、Debian でメンテナンスされている既存のパッチの見直しも必要です。

したがって、上記のような制約を踏まえながら、どのバージョンまであげていくのかを検討していかなければなりません。

6.4 Debian における Mozc を更新する方針

そこで、以下にような方針で更新にチャレンジすることにしました。

- 2.30 にいきなり上げるのは差分が大きすぎるので断念
- 2.29.5544.102 も新し目の abseil が必要になってくるので断念
- 2.29.5268.102 以降は Qt へ移行しないといけないのでやらない
- 2.29.5160.102 であればまだ Qt への移行は必要ないので 2.29.5160.102 をターゲットとする

Bazel の新しいバージョンが Debian unstable に投入されるのはまずないだろうし、GYP 対応のパッチを更新していくのが現実的と判断しました。

6.5 実際の更新のために実施したあれこれ

実際に更新するためには事前準備も含めていろいろありました。

- 既存のパッチをすべて見直しと状況を報告するバグを立てた[*40]
- パッチの見直しにともない発生した、2.29.5160.102 向けにあれこれ更新する作業
- debian/watch と debian/copyright を整備して、upstream のソースの repack 対応
- 2.29.5160.102 から削除されてしまった GYP 対応のコードをサルベージ

それから、いったん experimental にパッケージをアップロードして手元で検証する必要もありました。この原稿を書いている 12 月時点では、おそらく問題ないであろうと判断して delayed アップロードしてあります。[*41] 何事もなければ、数日以内に Debian unstable に 2.29.5160.102 が投入されるはずです。

6.6 さらなる更新への展望

Debian 13 (trixie) へ向けたフリーズがそろそろ始まるため、できるだけ Mozc も upstream に追従していきたいところではあるのですが、abseil や protobuf といった依存ライブラリーのより新しいバージョンが Debian unstable に降ってこないと 2.29.5268.102 以降に更新していくのは厳しいとみています。仮に降ってきたとしても、Qt への移行もすすめないといけないのでそのへんも考えないといけません。諸般の事情を鑑みると、このままいくと Debian 13 (trixie) では 2.29.5160.102 が採用されることになるでしょう。

[*40] https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=1085173
[*41] https://ftp-master.debian.org/deferred.html

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 オープンユーズロゴ 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.

『あんどきゅめんてっど 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 2024/06〜2024/11 まで。
内容は無保証、つっこみなどがあれば勉強会にて。

是非、Debian 勉強会に下記の QRCode から参加してください!

あんどきゅめんてっど でびあん 2024 年冬号

2024 年 12 月 30 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）

Debian 勉強会 connpass: https://debianjp.connpass.com
https://tokyodebian-team.pages.debian.net
https://wiki.debian.org/KansaiDebianMeeting