.Deb

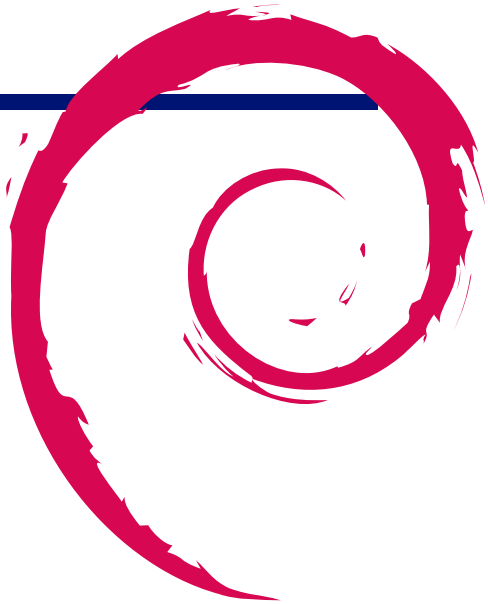
銀河系唯一のDebian専門誌

2021 年 5 月 15 日

go を使う

デビアン勉強会

目次



1　最近の Debian 関連のミーティング報告

杉本　典充

1.1　2021 年 4 月度 東京エリア・関西合同 Debian 勉強会

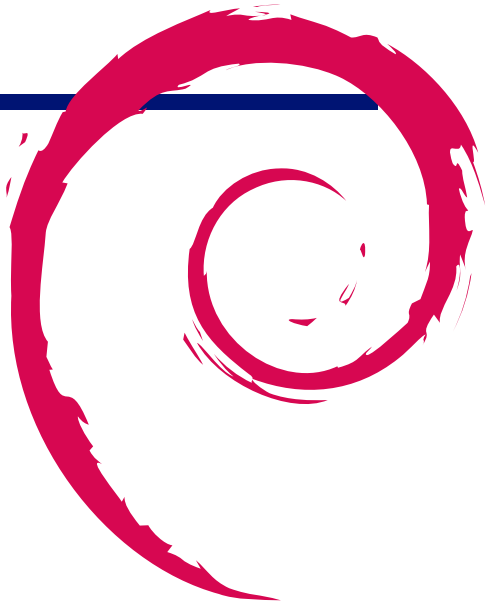
2021 年 4 月 17 日 (土) に東京エリア Debian 勉強会と関西 Debian 勉強会の合同でオンラインによる Debian 勉強会を開催しました。参加者は 11 名でした。

セミナーは西山さんによる「systemd 再入門」および参加者全員による「Debian パッケージの説明文を翻訳してみよう」を行いました。

「systemd 再入門」では、crontab の代わりに使える systemd-timer、user 権限で systemd からプログラムを実行する機能について説明がありました。systemd-timer は crontab の設定ではできないワンショットな設定も可能であり、多機能な優位点があります。ユーザ権限で systemd を使う機能は、”systemctl” に ”–user” オプションをつけて呼び出すこと、設定は ”.config/systemd/user/” のホームディレクトリ配下における仕様になっていると説明がありました。

「Debian パッケージの説明文を翻訳してみよう」では、Debian パッケージの説明文を翻訳システムを使って反映する仕組みを説明しました。`https://ddtp.debian.org/` というサイトでパッケージの説明文の翻訳データを入力してレビューするプロセスの説明がありました。

その後、Debian の最近の動向について情報交換を行いました。



2 事前課題

杉本　典充

今回の事前課題は以下です。

1. Debian で Go 言語を使ってプログラムを書いたことがありますか。
2. 2019 年にリリースした Debian 10 ”buster” のリリースノートは読みましたか（言語は問いません）。

2.1 dictoss

1. はい、書いたことがあります
2. 重要そうなページは読んだ, 読もうとしたが、英語のままの部分がありつらかった

2.2 tanykazy

1. はい、書いたことがあります
2. 重要そうなページは読んだ, 読んでいないが問題なく buster を使えた

2.3 NOKUBI Takatsugu (knok)

1. はい、書いたことがあります
2. 重要そうなページは読んだ

2.4 Hiroyuki Yamamoto (yama1066)

1. いいえ、書いたことはありませんが興味はあります。
2. 読んでいない

2.5 dancerj

1. はい、書いたことがあります
2. 重要そうなページは読んだ

2.6 kenhys

1. いいえ、書いたことはありません。
2. (回答なし)

2.7 yosuke_san

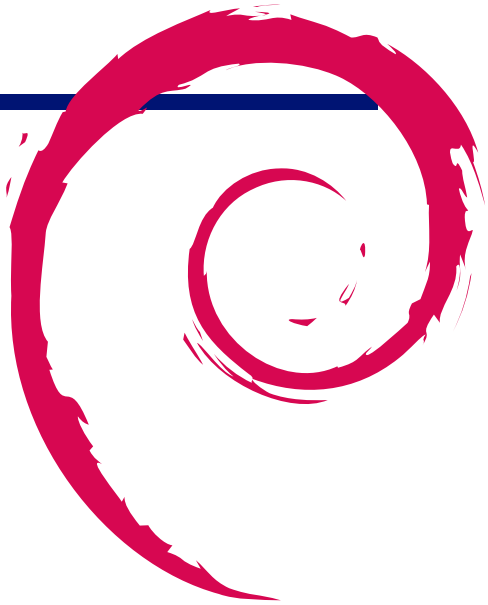
1. いいえ、書いたことはありませんが興味はあります。
2. 重要そうなページは読んだ

2.8 hisashi (hisamakijp)

1. いいえ、書いたことはありません。
2. 読もうとしたが、文量が多くて読むのをあきらめた

2.9 yy_y_ja_jp

1. はい、書いたことがあります
2. 重要そうなページは読んだ



3 Debian Trivia Quiz

username

Debian の昨今の話題についての Quiz です。

今回の出題範囲は debian-devel-announce@lists.debian.org や https://bits.debian.org/archives.html や https://micronews.debian.org/ などに投稿された内容からです。

問題 1. Bullseye 向けの Debian-installer のブロッカーバグは

☐ A http://bugs.debian.org/987441 を参照

☐ B http://bugs.debian.org/1 を参照

☐ C 強いて言えば https://lists.debian.org/debian-devel-announce/2021/05/msg00000.html に書いてある

問題 2. 5 月 3 日時点での Bullseye 向け RC バグの数はいくつだったか

☐ A 132

☐ B 0

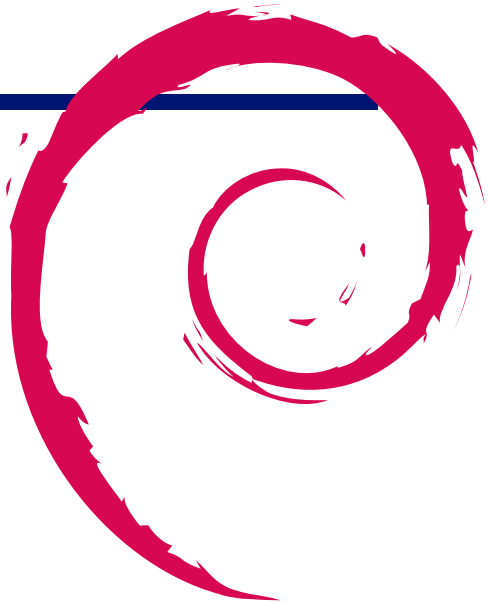
☐ C 100000

問題 3. unblock request を投げるときに何を含めるとよいか

☐ A なぜこれが入るべきかの長文演説

☐ B 自分の半生について語る日記

☐ C filterdiff で自動生成されたものを省略した diff



4 Debian における Go

上川純一

Go 言語を Debian で使ってみましょう。

まず入門するにはオンラインでのチュートリアルが充実しているので最初はローカルにインストールすることすら必要ないでしょう [4] [5]。しかし次第に本格的に使い始めようとすると手元にインストールしたくなるはずです。

4.1 Debian ユーザとして Go でプログラムを書いてみる

Go のインストール手順 [3] に従って公式パッケージをダウンロードして Tar を展開すると最新のバイナリがインストールできます。しかしあとからアンインストールするときとかに面倒ですしアップデートも自動ではありません。ここは Debian パッケージが使いたいなぁと思います。

Debian の公式パッケージは Buster では 2021 年 4 月現在 1.11 ですが、Buster backports にはもう少し新しい 1.14 があります。まず /etc/apt/sources.list に buster-backports を追加しましょう。[*1]

```
deb https://deb.debian.org/debian buster-backports main
```

そしておもむろにインストール

```
$ sudo apt update
$ sudo apt install -t buster-backports golang
$ sudo apt install -t buster-backports golang-mode  # もし Emacs ユーザなら
```

Go の追加のパッケージは Debian パッケージではなく Go の仕組みでインストールしましょう。Debian パッケージはビルド時の依存関係を満たすためだけにパッケージされているようです。理由は 2 つあり、

- ユーザが go get を使うことで go build などに必要な環境をユーザが設定することになる。
- ユーザ権限では書き込めないのと、.git ディレクトリが存在しないため通常の go get -u でのアップデートなどが不可能なためです。

以上の理由により Debian パッケージでは/usr/share/gocode/src/にソースがインストールされるのですが特に指定しないと Go のコンパイル時に利用してくれません。

たとえば sqlite3 を利用したいなと思ったら go get コマンドを利用してモジュールを ~/go 以下にインストールします。 [*2]

```
$ go get github.com/mattn/go-sqlite3
```

すると以下のようなプログラム (db.go) がコンパイルできるようになります。

[*1] 現在フリーズ中の次期安定版予定の bullseye だと 1.15 になっているようです。さらには sid には 1.16 パッケージがあるようです。

[*2] Debian パッケージだと golang-github-mattn-go-sqlite3-dev があるのだがそれはパッケージのビルド用にしか使えないようだ。

```
import (
        "database/sql"
        _ "github.com/mattn/go-sqlite3"
        "log"
)

func main() {
        db, err := sql.Open("sqlite3", "./ac.db")
        if err != nil {
                log.Fatal(err)
        }
        defer db.Close()
}
```

コンパイルは go build で実行します。ただビルドするだけじゃなくてそのまま実行までしたければ go run で実行できます。

```
$ go build db.go
$ ./db

$ go run db.go
```

4.1.1 Golang の Module システム

Golang の標準のモジュール管理システムは GOPATH が従来利用されていたのですが、1.11 から Go Modules[7] が提供され、1.14 からデフォルトに切り替わりました。

go.mod ファイルにモジュールメタデータが管理されています。[*3]。ホームディレクトリ以下の go/pkg 以下にモジュールがインストールされるようです。[*4]

Go Module を使う Go のコードを新規に開発するには、モジュールのディレクトリで go mod init で go.mod を作成します。

```
$ go mod init example.com/your/packagename
$ cat go.mod
module example.com/your/packagename

go 1.15

$ go test  # カレントディレクトリのモジュールのテストを走らせる
```

サブディレクトリのパッケージのときに import 文で example.com/your/packagename/subdir のように指定するとローカルのサブディレクトリを利用してくれます。

それ以外の名前を指定したら go build 時にダウンロードしてきて、go.mod に情報を追記してくれるようです。

4.1.2 Docker ユーザとしての Go

Docker で Debian で Go を使う場合には Dockerhub の Golang の Docker イメージ [8] を使うのが効率よくお手軽でしょう。

golang という名前でいくつかのバージョンが提供されておりデフォルトは Buster イメージのようです。golang:latest は２０２１年４月現在では golang:1.16.3-buster になっており docker pull や Dockerfile に golang と指定すると Debian の上に Golang をインストールしたイメージが入るようです。

たとえばこんな Dockerfile を書いて CI に利用しています。

```
FROM golang
COPY . .
RUN make
```

4.2 Debian 開発者としての Go パッケージ

Debian に Go でできたソフトウェアをパッケージとしていれたいときには追加で作業が必要になります。いちユーザとして利用していた場合との差異としてはポリシー準拠のために Debian で依存関係すべて含めてビルドできる

[*3] 以前はディレクトリ構成と GOPATH 環境変数で指定されたディレクトリで管理していました
[*4] 以前はホームディレクトリ以下の go/src

ようにする必要があるということです。つまり依存関係にあったものを go get でダウンロードしてくるのではなく Debian パッケージを利用します。

Debian にある Go パッケージ作成は pkg-go チームが中心となって行っているようです [6]、全てではないですが大半のパッケージはこのチームがやっているようです。

dh-golang を使って Debian パッケージのビルドは行うようです。dh-make-golang コマンドを使うと Debian パッケージの雛形を作成してくれるようです。

```
$ dh-make-golang make github.com/ncabatoff/fakescraper
2021/04/10 01:46:49 Starting "dh-make-golang v0.4.0 linux/amd64"
2021/04/10 01:46:49 Downloading "github.com/ncabatoff/fakescraper/..."
2021/04/10 01:46:50 Determining upstream version number
2021/04/10 01:46:50 Package version is "0.0~git20201102.4b37ba6"
2021/04/10 01:46:50 Determining dependencies
2021/04/10 01:46:50 Generating temp tarball as "/tmp/dh-make-golang230078837"
2021/04/10 01:46:50 Moving tempfile to "golang-github-ncabatoff-fakescraper_0.0~git20201102.4b37ba6.orig.tar.xz"
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:         git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:         git branch -m <name>
2021/04/10 01:46:50 Adding remote "origin" with URL "git@salsa.debian.org:go-team/packages/golang-github-ncabatoff-fakescraper.git"
2021/04/10 01:46:50 Adding remote "github" with URL "https://github.com/ncabatoff/fakescraper"
2021/04/10 01:46:50 Running "git fetch github"
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 14 (delta 0), reused 0 (delta 0), pack-reused 13
Unpacking objects: 100% (14/14), 3.26 KiB | 238.00 KiB/s, done.
From https://github.com/ncabatoff/fakescraper
 * [new branch]      master     -> github/master
2021/04/10 01:46:52 Could not determine license for "github.com/ncabatoff/fakescraper": GET https://api.github.com/repos/ncabatoff/fakescrape
2021/04/10 01:46:52 Setting debian/watch to track git HEAD
2021/04/10 01:46:52 Could not determine license for "github.com/ncabatoff/fakescraper": GET https://api.github.com/repos/ncabatoff/fakescrape
2021/04/10 01:46:52 Done!

Packaging successfully created in /tmp/kk/golang-github-ncabatoff-fakescraper
    Source: golang-github-ncabatoff-fakescraper
    Binary: golang-github-ncabatoff-fakescraper-dev

Resolve all TODOs in itp-golang-github-ncabatoff-fakescraper.txt, then email it out:
    /usr/sbin/sendmail -t < itp-golang-github-ncabatoff-fakescraper.txt

Resolve all the TODOs in debian/, find them using:
    grep -r TODO debian

To build the package, commit the packaging and use gbp buildpackage:
    git add debian && git commit -a -m 'Initial packaging'
    gbp buildpackage --git-pbuilder

To create the packaging git repository on salsa, use:
    dh-make-golang create-salsa-project golang-github-ncabatoff-fakescraper

Once you are happy with your packaging, push it to salsa using:
    gbp push

NOTE: Full upstream git history has been included as per pkg-go team's
      new workflow.  This feature is new and somewhat experimental,
      and all feedback are welcome!
      (For old behavior, use --upstream-git-history=false)

The upstream git history is being tracked with the remote named "github".
To upgrade to the latest upstream version, you may use something like:
    git fetch github          # note the latest tag or commit-ish
    uscan --report-status     # check we get the same tag or commit-ish
    gbp import-orig --sign-tags --uscan --upstream-vcs-tag=<commit-ish>
```

参考文献

[1] "The Go Programming Language" https://golang.org/

[2] "Documentation – The Go Programming Language", https://golang.org/doc

[3] https://golang.org/doc/install

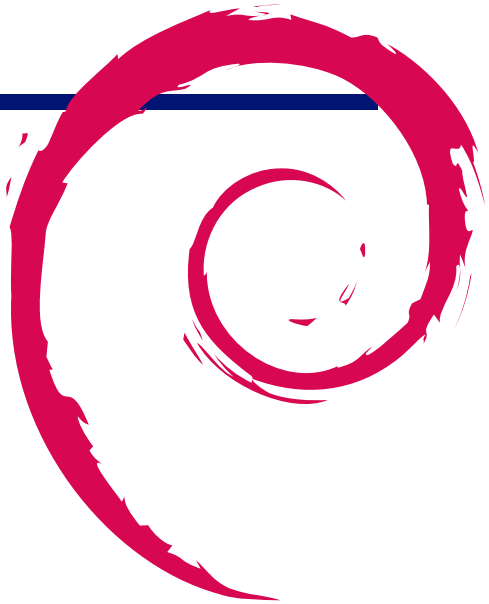
[4] "A Tour of Go" https://tour.golang.org/welcome/1

[5] https://go-tour-jp.appspot.com/welcome/1

[6] "Debian Go Packaging" https://go-team.pages.debian.net/packaging.html

[7] "'Go modules"' https://github.com/golang/go/wiki/Modules

[8] "golang Docker Official Images" https://hub.docker.com/_/golang



5 メモ

Debian 勉強会資料

2021 年 5 月 15 日　　初版第 1 刷発行

東京エリア Debian 勉強会 （編集・印刷・発行）