uopi4conf

mitsuhisaT

2021 年 08 月 10 日

Contents:

第 1 章　Ubuntu デスクトップを Raspberry Pi 4B 、 Pi400 にインストールする　3
- 1.1 事前準備 . . . 3
- 1.2 Ubuntu desktop for Pi4 の準備 . . . 4
- 1.3 初回起動 . . . 5
- 1.4 更新とアップグレード . . . 5

第 2 章　セキュリティについて　7

第 3 章　クラウド・ストレイジ　9

第 4 章　オープンソース版 Chrome　11
- 4.1 インストール . . . 11
- 4.2 設定 . . . 11
- 4.3 同期ができなくなった . . . 11

第 5 章　Atom エディタのインストールと設定　17
- 5.1 NODE.js のインストール (with *nvm*) . . . 17
- 5.2 Atom のインストール . . . 17
- 5.3 おすすめのパッケージ . . . 17

第 6 章　Python のインストール (with pyenv)　19

第 7 章　日本語入力について　21

第 8 章　ドキュメントについて　23
- 8.1 LibreOffice について . . . 23

第 9 章　Indices and tables　25

This project aim describing "how to setup developping environment on Raspberry Pi 4B and Pi400 using Ubuntu desktop".

このプロジェクトの目的は、「Ubuntu デスクトップを使用して Raspberry Pi4B および Pi400 で開発環境をセットアップする方法」について説明することです。

第 1 章

Ubuntu デスクトップを Raspberry Pi 4B 、Pi400 にインストールする

1.1 事前準備

- Raspberry Pi 4B の 4GB または 8GB モデル あるいは Raspberry Pi 400
- Raspberry Pi 4B 対応のケース
- USB 電源アダプター 5V/3A USB Type C
- 16GB 以上の microSD カード
- HDMI 接続可能なモニタまたは TV と HDMI ケーブル
- USB 接続のキーボード (Pi400 は不要) とマウス

警告: Raspberry Pi 4B と Raspberry Pi 400 の HDMI 接続端子は mini HDMI です。100 円ショップなどの変換アダプタを使って接続しても良いです。Raspberry Pi 400 は、2021 年 7 月 13 日に技術基準適合証明 (技適) の工事設計認証が完了し、7 月 29 日から 日本語キーボード版 が発売されています。技術基準適合証明 (技適) 番号 020-210106 の形式又は名称は「Raspberry Pi 400」となっているので、英語配列キーボード版も発売されると良いなぁ (切望)。

注釈: Raspberry Pi にはオンオフ SW が付いていません。SW 付きの Raspberry Pi 4B 対応の電源アダプターを用意しました。

注釈:

- 日本向け製品では無い 64GB の microSD カードを利用しています。説明文に日本語が無いだけで、同等のモノがとても安く入手できます。
- 持ち運び用に MISEDI 13.3 インチ 4K(UHD) を準備しました。モニター自体の性能は申し分無いのですが、標準設定だと、相対的に文字が小さく、読むの厳しいかな。家で 32 インチ 4K に接続している場合は、全く問題ないです。
- マウスは logicool M187 のブライトレッドを接続しています。公式ケースの赤と相性が良いです。ワイヤレス接続ですが、ドライバーなど必要無く、付属のレシーバーを USB ポートに差し込めばすぐに使えます。

注釈: 有線 LAN 接続する場合は、LAN ケーブル (Gigabit Ethernet を使う場合は Category 6 以上)

警告: Raspberry Pi 3B+ と Ubuntu Mate と Bluetooth マウスの組み合わせは、起動したけど使いもにならず、でした。

1.2 Ubuntu desktop for Pi4 の準備

Ubuntu Desktop Raspberry Pi から Raspberry Pi 用 Desktop (64-bit) のインストールイメージをダウンロードします。

注釈: Raspberry Pi 4 Model B の 4GB または 8GB モデル、Pi400 または CM4 (Compute Module 4) で稼働できます。

ダウンロードしたファイル (ISO イメージ) は balenaEtcher を利用して microSD に *Flash* します。

注釈: macOS では Homebrew を利用して `brew install balenaetcher` でインストールできます。

1.3 初回起動

1. microSD の挿入

(a) 前のステップで用意した microSD を Raspberry Pi に挿入する

2. ケーブル接続

(a) HDMI ケーブルを Raspberry Pi (micro HDMI) とモニターに接続する

(b) USB キーボードとマウスを Raspberry Pi に接続する

(c) 電源アダプターを Raspberry Pi (Type-C) に接続する

(d) 有線 LAN 接続する場合は、LAN ケーブルを接続する

3. ケーブル接続を確認

4. 電源アダプターをコンセントに挿す

5. セットアッププログラムが起動する

(a) ログインユーザの設定

(b) 言語設定： 英語を選択

(c) Wi-Fi ネットワークの設定

(d) タイムゾーンの設定

6. 待つ

(a) ログインプロンプトが表示されるまで待つ

(b) 5. a. で設定したパスワードを入力すると Ubuntu デスクトップが表示されます

1.4 更新とアップグレード

Show Applications (左下の 9 点のアイコン) の 2 ページの *Utilities* の 2 ページの *Terminal* をクリックし起動します。

まず、パッケージリストを次のコマンドを入力し更新します。キーボードで *sudo apt update* と入力し、最後に [Enter キー] を入力します。

```
sudo apt update
```

リストの更新が完了し、プロンプトが表示されたら、アップグレードをおこないます。

```
sudo apt upgrade
```

アップグレードされるアプリケーションやライブラリなどのリストとダウンロードの合計サイズが表示され、

```
Do you want to continue? [Y/n]
```

と表示されるので、*y* を入力します。更新されたパッケージをダウンロードし、その後にアップグレードが行われます。

警告: インストール直後のアップグレードなので、リスタートします。右上の電源ボタンをクリックし *Power Off/Logout* から *Restart...* を選択します。

注釈: 更新とアップグレードは、頻繁に実施することをお勧めします。不具合解消とセキュリティアップデートが実施できます。

第 2 章

セキュリティについて

Ubuntu Desktop のセキュリティについて記述する予定です。

第 3 章

クラウド・ストレイジ

Google Drive の利用と同期について記述する予定です。

第 4 章

オープンソース版 Chrome

Google Chrome WEB ブラウザのオープンソース版 Chromium について記述する予定です。

4.1 インストール

〇〇を利用してももちろんインストールできますが、

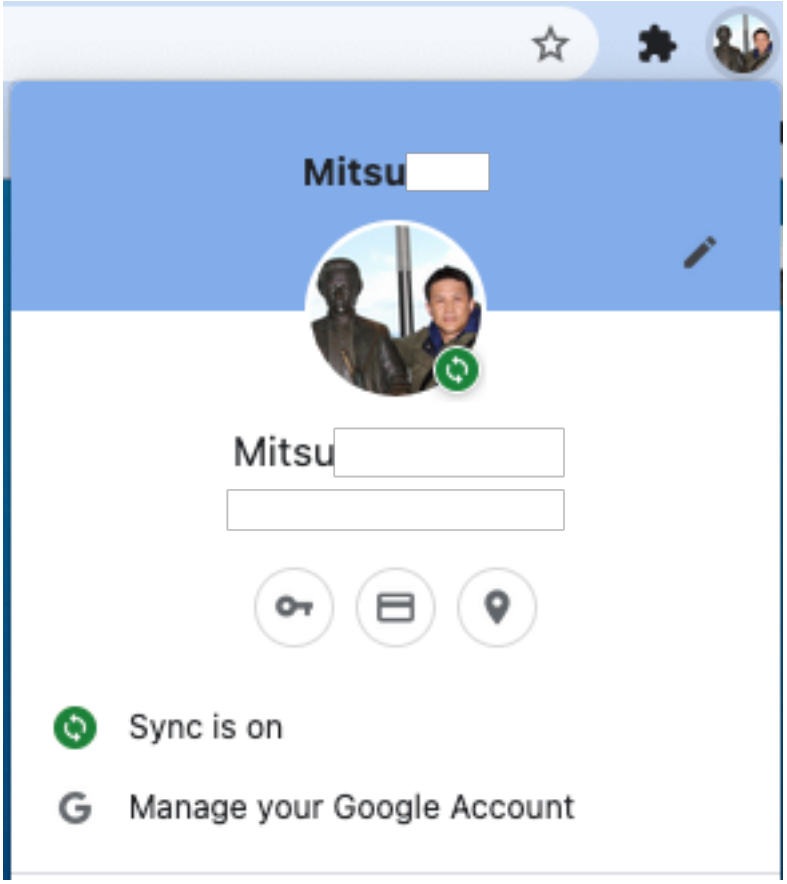
```
sudo apt install chromium
```

で、インストールしました。

4.2 設定

4.3 同期ができなくなった

Google Chrome のオープンソース版の Chromium で



これが出来なくなった。

2021 年 03 月 15 日から、 Chromium などでは、Google アカウントのデータ同期が出来なくなった = ブックマークなどの共有ができない。

Limiting Private Api availability in Chromium に、「2021 年 3 月 15 日から プライベート Chrome API に制限をかける」と有ります。

> We are limiting access to our private Chrome APIs starting on March 15, 2021.

解決策は、

- **手動の同期**
  - My Google Activity から Google Takeout を使う
- Chrome **ブラウザを利用する**
  - Linux 用は *Intel CPU* は、ダウンロード可能
  - Raspberry Pi が採用している ARM アーキテクチャ用は 無し
- Chromium **を自分用に作成する**
  - ソースコードの入手 Get the code

- **ビルド手順** Build Instructions
  - 必需品：16GB RAM 推奨、100GB の空きディスク、Git と Python v3

なので、かなり面倒ですが 手動 の動機を選びました。

4.3.1 intel Mac で ARM 用 Chromium をビルド

- Mac Build Instructions
- Chromium for Arm Macs
- Update on Google API usage in Chromium
- How GN handles cross-compiling
- API Keys

を参考にビルドしてみた。結果は、まだ Intel macOS で ARM64 Linux の Chromium は生成できなかった (^ ^;

Get the code　(ソースコードの取得)

fetch の 1 回目は、 *too many open files* で失敗した。

```
failed to read "/Users/mitsu/Develop/git/googlesource/chromium/src/components/ssl_errors
↪": fcntl: too many open files
Error: Command 'python3 src/testing/generate_location_tags.py --out src/testing/location_
↪tags.json' returned non-zero exit status 1 in /Users/mitsu/Develop/git/googlesource/
↪chromium
failed to read "/Users/mitsu/Develop/git/googlesource/chromium/src/components/ssl_errors
↪": fcntl: too many open files

Hook 'python3 src/testing/generate_location_tags.py --out src/testing/location_tags.json
↪' took 12.13 secs
Subprocess failed with return code 2.
```

Atom で開いているファイルを 5 個くらい閉じて、 2 回目で成功した。実行時間は 13 分 52 秒だった。

Setting up the build (ビルドの設定)

gn gen out/pi4arm64 を実行すると *vi* 系エディタが起動するので **Faster builds** の設定、クロスコンパイルの設定と、 API Keys の **Acquiring Keys** の手順を行い *client ID* 、 *client secret* も設定した。

Build Chromium

次のコマンドで Chromium をビルドする。

```
autoninja -C out/pi4arm64 chrome
```

81,658 ファイルのビルドが始まります。

実績値

- 1 回目 CCache 無効：5 時間 49 分 29 秒 (target_os 指定忘れ)
- **2 回目** CCache **有効：時間分秒** (target_os *linux*)
  - Intel mac では Linux/ARM64 は未だ NG

以下は、*out/pi4arm64/args.gn* の 2 回目の設定内容です。

```
# Set build arguments here. See `gn help buildargs`.
is_debug = false
```

(次のページに続く)

(前のページからの続き)

```
is_component_buils = true
symbol_level = 0
current_os = "mac"
current_cpu = "x64"
target_os = "linux"
target_cpu = "arm64"
google_api_key = "your_api_key"
google_default_client_id = "your_client_id"
google_default_client_secret = "your_client_secret"
```

ビルド前の設定内容の確認コマンドの結果

```
gn gen --check out/pi4arm64
ERROR copy_bundle_data tool not defined
The toolchain //build/toolchain/linux:clang_arm64
used by target //components/policy:chrome_manifest_bundle
doesn't define a "copy_bundle_data" tool.
```

create install package

ビルドが成功したら、以下のコマンドを実行すれば、deb パッケージが生成できたはず。

```
chrome/installer/linux/debian/build.sh out/pi4arm64/
```

第 5 章

Atom エディタのインストールと設定

Atom エディタのインストールと設定について記述する予定です。

5.1 NODE.js のインストール (with *nvm*)

5.2 Atom のインストール

5.3 おすすめのパッケージ

第 6 章

Python のインストール (with pyenv)

Python のインストールと設定にについて記述する予定です。

第 7 章

日本語入力について

日本語入力の設定について記述する予定です。

第 8 章

ドキュメントについて

ドキュメンテイションについて

8.1 LibreOffice について

オープンソースのオフィススイート *LibreOffice* について簡単に紹介します。

第 9 章

Indices and tables

- genindex
- modindex
- search