# 概要

## このドキュメントの目的

Raspberry Pi OS の詳細情報などに重点を置く。

但し，別ドキュメントRaspberryPi と重複してしまう情報もどうしてもでてくる。

## 基本情報

・ラズパイの基本OS，Raspberry Pi OS はDebian系なので，Debian，Ubuntu の情報が使える場合が多い。

　Web検索する時，”ラズパイ” や “Raspberry Pi OS”　で情報が見つからなかったら，”Ubuntu” などを試す。

Raspberry Pi OS（Raspberry Pi OS）のバージョン

Buster 10

Stretch 9

Raspberry Pi OS（Raspberry Pi OS）の情報

GUIはX Window Systemを採用。

set\_mode

## Raspberry Pi OSのインストール

### インストール手順

microSDの用意

運営の推奨は**SD Card Formatter**でフォーマットされたmicroSD**。**SDカードの速度性能を最大限に引き出せる。ダウンロードサイト：<https://www.sdcard.org/jp/downloads/formatter/>

（補足）

・microSDはFAT32，パーティションが無い状態にする。

・容量は8/16/32G のいずれかが良い。

・実はusbに書き込んでのusbブートも可能。

OS（ディストリビューション）のダウンロード

最新版は公式サイトで落とせる：Software → Raspberry Pi OS

[https://www.raspberrypi.org](https://www.raspberrypi.org/)

旧バージョンは基本的にミラーサーバーで落とす（どちらのサーバーでも良い）

<http://ftp.jaist.ac.jp/pub/raspberrypi/>

[http://ftp.yz.yamagata-u.ac.jp/pub/linux/Raspberry Pi OS/](http://ftp.yz.yamagata-u.ac.jp/pub/linux/raspbian/)

OS（ディストリビューション）の書き込み

2020年3月に公式の書き込みツールが作成された。<https://www.raspberrypi.org/downloads/>

もしくはLinux機があれば「DDコマンド」で書き込む事もできる。

以下はOSイメージを書き込むアプリの旧情報。Windows, Mac 用。

**Etcher** ：割と新しいアプリ。

Win32DiskImager ：Web上ではそこそこ人気。

DD for windows R2 ：SDがきちんとフォーマットされてないと動かないらしい。

※どのアプリでも書き込み自体は可能。あとは好みとパソコンとの相性。

補足：書込みの際にSDカードはアンマウントされるので，書込み直後のSDカードはそのまま取り外せる。

むしろ既にアンマウントされている為に「安全に取り外す」ができない。

### クライアント側で用意した方が良いアプリ

　※クライアント：サーバーを利用する側のコンピュータ。 ≒Windows，Macやスマホなどの事

#### リモート通信(SSH)

・OpenSSH

Windows 10には標準で入っている。入っていなくても無料なのでダウンロード。

使い方

※ラズパイ側で先に，hostname でSSID，又はifconfig/ip addr showなどでIPアドレスを確認しておく。

# Command Prompt

**::** ssh ログインユーザー名@ホストIP/SSID　　でログインできる。

**::** 例

ssh pi@192.168.1.15 **::** ユーザー名@ IPアドレスかDNS名。

exit **::** ssh接続の終了

ssh -p 1234 pi@192.168.1.15 **::** ポート番号を指定する場合　1234がポート番号

**::** 秘密鍵を用いたログイン方法もある

**::** 保留

（補足）

SSHにはTera Term , Putty, Poderosa などもあるらしい。

telnet ：一時期リモートアクセスの王道だった気がしたが，もうあまり使われていないとか

Windowsでの有効化 → Windowsの機能の有効化または無効化：Telnet クライアント

#### ファイル転送(SCP)

・WinSCP

WindowsからSFTPでファイルを転送するアプリ。要するに，遠方のラズパイにもデータのアップロード／ダウンロードができる。使うにはラズパイ側でSSHを有効にする必要がある。

### VirtualBoxでRaspberry Pi OSを使う場合。（編集中）

仮想マシンをother linux **64bit**で作成

設定 → システム → プロセッサー → プロセッサー数を２にする。

Raspberry Pi OSのインストールISOをマウント

ブートローダ部分の設定が必要らしい？？

<http://app-review.poox.xyz/archives/1625>

### その他の情報

・NOOBS を使ってインストールする手もあるらしい。

以下の中から一つを選んでインストールできる（但しRaspberry Pi OS単体のイメージよりはmicroSDを圧迫する）：

Raspberry Pi OS，Ubuntu MATE，Windows 10 IOT Core

LibreELEC，OSMC，Recalbox，RISC OS ，Screenly OSE など（バージョンにより異なる？）

・ベンチマーク

（コンピュータの能力を測って，数値化する事。これは興味があれば，で良い）

UnixBench を利用する場合

git clone https://github.com/kdlucas/byte-unixbench # cloneを作成

cd byte-unixbench/UnixBench # ディレクトリ移動（移動しておく必要がある）

./Run # 実行

（参考）[GitHub](#_GitHub)

他にも，Passmark や i-nex （CPU-Z のLinux版）の情報もあるが，未だ試していない。

# Raspberry Pi OS初心者向き情報

## SD Card Copier

 →　アクセサリ　→ SD Card Copier　（Raspberry Pi OS のデスクトップ版のみ）

microSD カードのコピー機能。特に開発実験などをしていると，その途中で不具合が生じてしまう事もある。この機能でmicroSDを丸ごとコピーする事でシステムのバックアップを取る事ができる。

（補足）

・microSDは２，３枚あった方が便利。

・dd というコマンドを使ってバックアップファイルを作成する事もできる。[ddを使ったバックアップ](#_DD_を使用する場合（CUI）)

## ドライブの取り外し

タスクバーのボタンから取り外しが可能。

（左図を参照）

※SD Card Copierを使う時は，念の為USBメモリなどを取り外しておいた方が良い。

（間違えてUSBメモリを初期化しない為）

## タスクバーの位置

Raspberry Pi OS のタスクバーの位置はデフォルトでは上になっているが，この位置は変更可能。

タスクバーを右クリック → パネルの設定：[ジオメトリ] タブの 「位置」

## ファイルのアクセス権

Linuxのファイルシステムでは，各ファイルに「アクセス権」が設定されている。誰でもアクセスできるファイルもあれば，「super user」 しかアクセスできないファイルもある。これはWindowsなどでは馴染みのないシステムで，少々とっつきにくい。

なお，アクセス権はTerminalのlsコマンドで確認できるので，色々なディレクトリで試してみると良い。

#Terminal （例）

ls -l # この例ではhomeディレクトリ内のファイルの一覧が表示される。

# ※ -l オプションはリスト形式の詳細表示 ※homeディレクトリ ≒ windowsのマイドキュメント

ls -l /etc/apt　# ディレクトリ指定して実行。　（もしくはcdで移動してから ls -l でも良い。）

>> d**rwxr-xr-x** 2 pi pi 4096 1月 1 15:40 Desktop

# この場合，所有者に［読み，書き，実行］，所属グループに［読み，実行］，その他に［読み，実行］の権限。

# 更に，この「Desktop」はディレクトリであり，所有者はpi，所属グループもpiである事を示している。

（参考）<https://qiita.com/shisama/items/5f4c4fa768642aad9e06>

ファイルシステムに関する予備知識

Windowsで使っていたUSBメモリがLinuxでは使えない事がある。これはディスクのフォーマットが違う為で，具体的には，近年のWindowsの標準NTFSはLinuxでは使用できないし，Linux標準のext4やext3もWindowsでは使用できない。Windows, Mac, Linux共通で使うドライブはexFATでフォーマットする。

*※FAT32もLinuxで使用可能ではあるが，FAT32には１ファイル4GBまで，という制限がある。*

## Terminal

Linux(Raspberry Pi OS)　ではコマンドを頻繁に利用する事になる。

コマンドのアプリはTerminal と言い，Ctrl＋Alt＋Tで起動できる。

もしくは，タスクバーからでも起動できる。（左図参照）

コマンドはパラメータ（引数）と組み合わせて使う事が多い。コマンドとパラメータの間には半角スペースを入れる。

使用例：ディレクトリ（フォルダ）の新規作成

mkdir␣/home/pi/test\_dir # mkdir がコマンド。

↑スペース

# 【 home/pi/test\_dir 】 がパラメータ。この場合，新規のディレクトリ名を指定している。

なお，コマンドアプリはコマンドプロンプト，コンソールなどとも呼ばれる。

Linuxのコマンドでは，ファイル名などは大文字小文字を区別するので注意する。

Terminal で使用される記号の説明（一部抜粋）

/ ディレクトリ（フォルダ）の区切り文字。スラッシュでなくバックスラッシュな点に注意。

\(バックスラッシュ)で終わる行は、次の行へ継続

./ カレントディレクトリ。（現在の作業ディレクトリの事）

../ 一階層上

~ ホームディレクトリ。 Windowsで言うマイドキュメントの様なもの。初期値では /home/pi

# コメント。この部分はコマンドとして処理されない。

コマンドの具体例：

# パッケージのインストール（linux ではアプリの事をパッケージと言う）

sudo apt install fcitx-mozc # fcitx-mozc　というパッケージをインストールする

sudo apt install postgresql postgresql-contrib　# 空白で区切る事で２つ以上を指定

# アプリの起動

geany /home/pi/src/test.py # プログラム開発アプリ「geany」で指定ファイルを開く

オプション（　とヘルプ ）

「オプション」で挙動の細かい指定ができる。

オプションにはハイフン１つ型(-h)　とハイフン２つ型 (--help) がある。

#Terminal

usermod -h # usermod –help でも良い。　⇒ ヘルプを見る；各オプションを見る事ができる。

ls --help # ls コマンドの場合，ヘルプ表示にはハイフン２つ型を用いる。

（補足）ハイフン１つ型が伝統的に使われていたが，分かりやすくする為にハイフン２つ型が後出で追加されたらしい。

### sudoについて

Linuxのユーザーの中には，様々な設定を変更する事ができる管理者，super user というものが存在する。

Windowsで言うadministratorの事。

sudoは一時的にsuper userの権限でコマンドを実行する時に付ける。 sudo = substitute user do

### nanoの使い方（抜粋）

Ctrl + X で編集を終了 → Y 保存；N 保存しない；C　終了のキャンセル

範囲を選んでからCtrl+K → 切り取り ；範囲選ばずにCtrl+K → 一行切り取り ；ペースト Ctrl + U

（詳細）[nano](#_nano)

その他

・sudo su super user に完全に切り替えて実行できる。sudoつけても実行できない場合はこれを試す。

# 初期設定

## Step0：事前設定

1. ある程度慣れた人向き。初心者にはおそらく無理なので，初心者は読み飛ばす。
2. メモ：イメージの書き込みの一例（Linux機が必要）
3. sudo dd if=/home/user/\*\*\*.img of=/dev/sdb bs=1M status=progress
4. unzip -p ./\*\*\*.zip | dd of=/dev/sdb bs=1M status=progress # 展開しながら書く
5. unxz -c ./\*\*\*.img.xz | dd of=/dev/sdb bs=1M status=progress # xzの場合

RaspberryPiOS のイメージが書き終わると，SDカードには２つのパーティションが作成される。

rootfsパーティションはext4なので，操作に別のLinux機が必要。

bootパーティションはFAT32なので，Windowsでも見られる。

### SSHの有効化

1. /boot ディレクトリに，拡張子なしのssh という**空ファイル**を作る。（ディレクトリでも良いらしい）
2. copy nul ssh :: Windows コマンドプロンプトの場合
3. touch ssh # LinuxやMacの場合

（補足）初期のユーザー情報 id:pi pass:raspberry

### /boot/raspi-config

1. raspi-configの設定ファイル /boot/config.txt を編集する事で，各種設定を先に行える。
2. dtparam=spi=on # SPIをONにする。
3. （補足）このconfig.txtはBIOSの代わりらしい。

次ページへ続く

〜Step0：事前設定　つづき

### その他設定ファイル(/rootfs)

1. （すべて管理者権限が必要）
2. rootfs/etc/wpa\_supplicant/wpa\_supplicant.conf wifi設定（有線LANの場合は不要）
3. rootfs /etc /dhcpcd.conf IPアドレス固定
4. rootfs /etc/apt/sources.list リポジトリ設定

wifi設定（有線LANの場合は不要）

1. [ /etc/wpa\_supplicant/wpa\_supplicant.conf ]
2. country=JP # Wifiの国
3. network={
4. ssid=”SSID\_NAME”
5. psk=”password”
6. key\_mgmt=WPA-PSK # 暗号化の種類を選ぶ。大体 WPA-PSKで良い。
7. }

wpa\_supplicant.conf　は/bootに置くことで自動的に/etc/ の物が更新されるらしい。

ファイル作成のWebツールが存在する　https://mascii.github.io/wpa-supplicant-conf-tool/

IPアドレス固定（任意）

1. [ /etc /dhcpcd.conf ]　（※設定ファイルの中にも例文がある）
2. interface eth0 # 無線の場合はwlan0
3. static ip\_address=192.168.1.15/24 # （一例） gatewayの設定は無くても動くかも
4. 補足：mDNS（Bonjour）が使える環境ならIPアドレスが不明でも
5. ssh pi@raspberrypi.local # パスワードはraspberry
6. システム復帰用sdカードを作る際に注意する事
7. ・他人に渡す際はwifiのSSID，プロバイダのpppの接続情報などの**個人情報を必ず削除**。
8. ・自動ログインを有効にしておく。もしくは，ユーザー名とパスワードが分かるようにする。
9. ・wifi, blue tooth, VNC，場合によってはsshを必要に応じてONに。（単純に便利だから）
   * 1. ・シリアル通信を使ったログイン方法もあるが，未だ試していない 20.08.25

## Step1：最初の設定

Raspberry Pi OS を起動すると，初期設定用ダイアログで以下の項目が設定できる。

国と言語とタイムゾーン；パスワード設定；Web接続；　など

一部初期設定用ダイアログの内容と重複するが，最低限の設定項目を順を追って記す。

### インターネットへの接続

1. GUI（デスクトップ）の場合
2. Raspberry Pi OSの更新に先立ち，ラズパイでインターネットに接続しておく。wifiで接続するのが楽。
3. タスクバーの　　　　　をクリック → 接続可能なネットワーク名(SSID)を選ぶ。
4. なお，Raspberry Pi OS のデフォルトのWebブラウザはChromium といい，Google Chrome のベースになったアプリ。タスクバーからも起動可能（左図参照）

CUI（コマンド）の場合

sudo ifdown wlan0 # 今接続しているようであれば，一旦切っておく。

sudo iwconfig wlan0 essid 接続先のSSID key s:接続先のキー

※上記はWifiの時で，有線LANの場合は挿すだけで良い。

ブルートゥースの無効化

ブルートゥースは実はかなりのシステム的負荷になっている。使わないならオフして置く。

[ /boot/config.txt]

dtoverlay=disable-bt

### キーボードの設定

# Terminal

sudo raspi-config # ラズパイの設定用コマンドraspi-config を利用する

# → 5. Localisation Options　→ L3 Keyboard

# 状況によりけりだが，大体の場合で【 モデル：標準105　　レイアウト：OADG 109A 】

Keyboard Model : **Generic 105-Key (Intl) PC**

Keyboard layout : Other -> Japanese -> **Japanese(OADG 109A)**

Key to function as AltGr : The default for keyboard layout

Compose key : No compose key

※raspi-config の実行にはパスワードが必要。

※Web 接続が必要

※初期ユーザーはpiでパスワードはraspberryになっている。Raspberry Pi OS の初期設定画面でパスワードを設定していない場合にはraspberryが適用されている点に注意。

ついでに5. Localisation Optionsの他の設定をしておく：

L1 Locale

ja\_JP.UTF-8 UTF-8を選択してスペースで決定→ Tabを押下 → <Ok>

デフォルトは ja\_JP.UTF-8 を選択

ロケールの参考： en\_GB.UTF-8 UTF-8 ja\_JP.EUC-JP EUC-JP ja\_JP.UTF-8 UTF-8

L2 Timezone → Asia → Tokyo

L4 WLAN Country Set → JP Japan

※ここで再起動が必要

## Step2：リポジトリの変更

初心者は読み飛ばしても良い。変更しなくても使えないことは無い。

OSの更新，アプリのダウンロードなどでRaspberry Pi OSサーバー（リポジトリ）には頻繁にアクセスするので，下のurl先のリストのうち，近場のものに変えておく。

http://www.Raspbian.org/RaspbianMirrors

設定ファイルは/etc/apt/sources.list で，編集には管理者権限が必要。

Terminal

sudo nano /etc/apt/sources.list # 設定ファイルの編集

# ※ファイルパスの最初のバックスラッシュは忘れがち。注意。

# このコマンドは「nanoというアプリを起動 + 編集するファイル指定」　の意

# nano：コンソール（Terminal）で動くテキストエディタ。desktop環境ならleafpad を使っても良い。

[ /etc/apt/sources.list ]

# 設定ファイルの書き方の一例　　　日本のJAISTのサーバーを指定

deb http://ftp.jaist.ac.jp/Raspberry Pi OS stretch main contrib non-free rpi

# 元の設定はコメントアウトするか，今回追加したものより下に行くようにしておく。

（注意）source.listを指定したのに何も書いていない空の表示だった場合，**ファイルパスの入力ミス**があった可能性が高い。まだ存在しないファイル名を指定した場合は，その名前でファイルを新規作成となる。

（補足）**タイピングが得意でない人は**対象ファイルのパスをコピーすると良い。具体的に上のsources.listの場合は，ファイルマネージャでetc → apt と進みsources.list を右クリックして**「パスをコピーする」**を選択。その後，Terminal上で右クリックして貼り付け，又はCtrl+Shift+V でファイルパスを張り付けられる。

## Step3：更新＋日本語入力

**ここは必須。**

ターミナルを開く。以下の３つの内のいずれか。（どれでも良い）

 → アクセサリ → LXTerminal

Ctrl + Alt + T

　LXTerminal

### 更新

Terminal

apt update # パッケージを管理しているリストの取得。※まだRaspberry Pi OSは更新されていない。

apt upgrade # 取得したリストに基づいて実際に更新する。-y をつけると自動的に「はい」に。

apt autoremove # 更新に伴い，必要なくなったパッケージが削除される。

apt autoclean # キャッシュされ，システムにはインストールされていない「.deb」ファイルを削除。

（参考）<https://dbpro.xyz/1317>

（公式）<https://www.debian.org/doc/manuals/debian-reference/ch02.ja.html>

不要アプリのアンインストール（必要に応じて）

Terminal

sudo apt remove パッケージ名 # パッケージの削除

sudo apt remove wolfram-engine # 具体例：Wolfram を削除（設定ファイルは残る）

sudo apt --purge remove　パッケージ名 # 設定ファイルも削除　apt purge パッケージ名　と同等

### 日本語入力

Raspberry Pi OS は標準では日本語入力ができないので，日本語入力用のパッケージ，fctic-mozcをインストールする。

Terminal:

sudo apt install fcitx-mozc

日本語用フォントのダウンロード 2021-08-06 追加情報

Terminal:

sudo apt install fonts-ipafont fonts-ipaexfont

## Step4：SSH

SSHを有効にするとクライアント機（Windows, Mac等）を使ってラズパイの操作，設定ができる。

ラズパイ用のモニタが用意できない時などに便利。

必要なければ読み飛ばしても良い。

通常，サーバーが手に届く範囲内に置いてあり，直接操作できる事は少ない。特にレンタルサーバーを利用する場合などは，SSHを利用して遠方のサーバーの設定をする事の方が多いと思われる。練習と思って一度は経験しておいた方が良い。

### IPアドレスの確認

Terminal

ifconfig # wifi ならwlan0 の，有線ならeth0 を見る （例） inet 192.168.1.15

ip addr show # こちらの方が正式コマンドらしい。但し少し見にくい。

eth0 又は wlan0 にIPアドレスが割り振られている事を確認する。　無かったら ⇒ [IPアドレス固定](#_IPアドレス固定)

※lo inet 127.0.0.1；このアドレスは用途が限定されていて，今回は使えない。（localhostとかloopback とか呼ばれる。）

SSHの有効化

Terminal

sudo raspi-config # ラズパイの設定用コマンドraspi-config を利用する

# → 5. Interfacing Options　→ P2 SSH → to be enable? に <はい>

⇒ ここまで出来たら，クライアント側で　ssh pi@192.168.1.15 などとして接続してみる。（ユーザー名@IPアドレス）

ラズパイを再インストールしたりすると，クライアント側でホスト（この場合，ラズパイの事）情報を更新しないとSSH接続できなくなる。

その場合は以下のファイルの対象部分を削除する。（両方とも隠しディレクトリな点に注意！）

linux： $HOME/.ssh/known\_hosts

windows： users\[ユーザー名]\.ssh\known\_hosts

（補足）

SSH通信はPort22(TCP) と決まっている。これは一見便利なようだが，当然悪意あるユーザーも22番ポートを狙ってくる。知識がついてきたら，グローバルネットワーク(WAN)にラズパイを置くことを想定してSSHのポート番号を変えてみる。（但しWANにラズパイを置くことは少ない気がするが）

# /etc/ssh/sshd\_config

Port 22 # ここを変更。49152～65535あたりで好きなポート番号を選んで設定

### IPアドレス固定

SSH接続の為に，IPアドレスの固定が必要になった場合

[ /etc /dhcpcd.conf ]　（※設定ファイルの中にも例文がある）

interface eth0 # eth0 は有線，wlan0 は無線（wifi）

static ip\_address=192.168.1.15/24 # IPアドレス。当然，パソコンのIPアドレスと同じネットワークにする。

# パソコンとのラズパイの間にルーターがあれば，以下３行も設定（不要かも？）

static routers=192.168.1.254 #　ルーターのIPアドレス

static domain\_name\_servers=192.168.1.254

static broadcast 192.168.1.255 # 単にbroadcast でもいいらしい。この行は無くても良い。

Terminal

sudo service dhcpcd reload # IPアドレスを変更したら，サービスを再起動する

※domain name serverは必ずしも同じではないが，だいたいルーターのIPアドレスとなる。

### IPアドレスの設定方法

ラズパイはネットワーク機器なので，どうしてもIPアドレスの知識が求められる。

知識が無いと，ラズパイのIPアドレス192.168.1.15/24 に対してWindows(クライアント)はどのようなアドレスを設定すべきかが分からないかもしれない。

説明は大掛かりになってしまうので，分かりやすそうなサイトのリンクを貼っておく。

（参考）https://qiita.com/mogulla3/items/efb4c9328d82d24d98e6

どうしても理解できない場合は，wifiに対応したルーターを使用し，そのルータのwifiにラズパイとWindows(クライアント)を接続すると良い。通常ルーターがDHCPでIPアドレスを**自動で割り振る**ので，あとは各自のIPアドレスを調べるだけで良い。

※その場合，ラズパイのIPアドレスはifconfig で表示されるwlan0 のものを使用する。

## Step5：残りの設定

### ユーザー名変更

デフォルトのユーザー名（pi）はセキュリティ的に不安。場合にもよるが変更しておく。

注意点

対象ユーザーのログオフ，またはオートログイン設定の解除が必要。

GUI：設定 → Raspberrypiの設定→「システム」：現在のユーザーとしてログインする　のチェックを外す。

CUI：raspi-config → System Options → S5 Boot / Auto Login

ユーザー名変更方法の一例 [ SSHでクライアント側からの操作を想定している（※１）]

# Terminal

sudo useradd -M tmp # -M オプション；ホームディレクトリを作らない

sudo gpasswd -a tmp sudo # tmpユーザをsudoグループに追加

sudo passwd tmp # パスワード変更　※パスワードの入力文字は表示されないが気にしなくて良い

exit # 今作ったtmpユーザーでラズパイにログインする為にpiはログアウト。

# tmp でログインし直す

sudo usermod -l newpi pi # usermod -lでユーザ名をpiからnewpiに変更

sudo usermod -d /home/newpi -m newpi # usermod -dでホームディレクトリ変更

sudo groupmod -n newpi pi # groupmod -nでpiグループをnewpiグループに変更

sudo passwd newpi # newpiユーザーのパスワードを変更（後でも可）

# newpi でログインしなおす # 新パスワード，またはpiユーザーのパスワード

sudo userdel tmp # tmpユーザーの削除

※tmp ユーザーでログイン中は日本語キーボード設定ができず，Backspaceが使えない事があるので打ち間違いに注意。

（参考）<https://jyn.jp/raspberrypi-username-change/>

おまけ：SSHでなく，ラズパイで直接操作したい場合

Consoleの日本語表示パッケージの追加

sudo apt install jfbterm # consoleの日本語表示パッケージをインストール

jbterm # jbtermを起動する。

rc.localを用いてラズパイ起動時に自動起動する事もできる。

・デフォルトユーザー（pi） の自動ログインを解除。

上の方法か、raspi-config → 3.Boot Options → B1 Desktop/CLI → B1　Console ; requiring to loginでも可。要再起動。

・デスクトップモードで起動した際はCtrl+F3 でコンソールモードに移行する（Ctrl+F7で戻れる）

・なにかしらのユーザー（piで良い）でコンソールモード側でログイン（デスクトップモード側ではログインしない事）

⇒ あとは上の手順でユーザー名を変更する。

### raspi-config

SSHの有効化以外にraspi-config でやっておくべきこと。

・パスワード変更 1 Change User Password

・ホスト名変更 2 Network Options – N2 Hostname

・ファイルシステムの拡張 7 Advanced Options – A1 Expand Filesystem SDの全容量が使える。

⇒ 特に[dd を使ったバックアップ](#_DD_を使用する場合（CUI）) でファイルシステムが縮小されている場合に便利

・configの更新 8 Update

### 時刻設定

（注意）ラズパイは標準では時計機能がないので，電源OFF時している間，時間がずれ続ける。

（補足）インターネットに接続している場合は，一定時間間隔で時刻は自動補正される。

sudo date --set=" 2018-06-29 18:40:30" # 手動で合わせる場合

sudo systemctl restart systemd-timesyncd # 自動補正（NTPサービスのリロード）の場合

### 解像度の設定

モニタに合わせて解像度は自動で変更されるが，それでも見にくい事はある。

tvservice -s # 現在の解像度などを確認

 → アクセサリ → 設定 → Raspberry Piの設定

なお，/boot/config.txtでも設定ができる。

（参考）〔英語〕[https://elinux.org/RPiconfig#Video\_mode\_options](https://elinux.org/RPiconfig" \l "Video_mode_options)

### バックアップ

初期設定が終わった段階でバックアップを取っておくと良い。

（参考）[バックアップ](#_バックアップ_1)

# 詳細

## OS設定

### PATHの初期値

稀に誤って上書きしてしまう事がある為，ここに初期値を残す。

/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/local/games:/usr/games

※但し，再起動すると復活する。逆に永続化したい場合には~/.profileなどを利用する（後述）。

解説

/usr/local/sbin:

/usr/local/bin:

/usr/sbin:

/usr/bin:

/sbin:

/bin:

/usr/local/games:

/usr/games

~/.profile

ユーザーログイン時にBourne互換シェル（shなど）で実行されるスクリプト

if [ -d “$

## OSによる違い

### Bookworm

## セキュリティ

# セキュリティをたかめる

https://qiita.com/yyano/items/126cff2e3c49b0006c43

## 基本操作

|  |  |
| --- | --- |
| GUIからCUIに切り替える | Ctrl+Alt+F3 |
| CUIからGUIに切り替える | Ctrl+Alt+F7 |

### フォント

フォントは色々選択できるらしい（深くは調べていない）

# 日本語用フォントのダウンロード（効果なかった）　NOmore TOfu の略らしい。

sudo apt install fonts-noto

sudo apt install fonts-ipafont fonts-ipaexfont # その他で割と人気なフォント

sudo apt-cache search font japanese |grep noto

スペースで選択できるので，ja\_JP.EUC-JP EUC-JPとja\_JP.UTF-8 UTF-8」の２つにチェックを入れる。

デフォルトで使う文字コードにはja\_JP.UTF-8 を選択

apt install jfbterm # jfbterm はexitするとハングアップするので，fbtermをインストールする

### スクリーンがスリープしないようにする

# /etc/xdg/lxsession/LXDE-pi/autostart @xset s off

@xset s noblank

@xset -dpms

# /etc/lightdm/lightdm.conf

[SeatDefaults]

xserver-command=X -s 0 -dpms

情報元：<https://nabe-hiro.hatenablog.com/entry/2020/03/05/134722>

### 外付けデバイス

ls -l /media/pi # 接続された外付けデバイスの名前を確認（リストで表示）　※当然cdでフォルダ移動もできる。

sudo umount　/media/pi/0EEC-0231 # デバイスのアンマウント　※コマンドがunmount でない事に注意

USBメモリ

ls -l /media/[ユーザー名] # 現在マウントされているusbメモリのリストを表示

（補足）Raspberry Pi OSがデスクトップモードの時は、USBメモリを指すだけで自動的にマウントされるが、コンソールモードの時はmount コマンドを使って自分でマウントする必要がある。

### ブルートゥース

#### bluetoothctl

Terminalを用いてブルートゥースの設定やペアリングを行う。

※デスクトップモードであれば、タスクバーからのペアリングも可能。

sudo bluetoothctl # 起動（bluezというパッケージが最初からインストールされている）

##### 手順（ペアリング）

sudo bluetoothctl # 起動

>> [NEW] Controller XX:XX:XX:XX:XX:XX raspberrypi [default]

[bluetooth]# show # 必要に応じて状況確認

[bluetooth]# power on # 通常onだが、offになっている場合がある。

# sudo rfkill unblock bluettoth # power onがblocked された場合。

[bluetooth]# discoverable on # 通常off　ペアリングの為、一時的にon

# power, pairable, discoverableがすべて yes(on) が必要

[bluetooth]# scan on # 周囲のブルートゥースデバイスのスキャン

[bluetooth]# devices # 接続デバイスのMACアドレスを確認

>> Device 00:00:00:00:00:00 nexus7 #

[bluetooth]# pair 00:00:00:00:00:00 # ペアリング（Connected が yes になった後に直ぐに noになる）

[bluetooth]# trust 00:00:00:00:00:00 # 認証を行う（Connected が再び yes になる）

##### 各コマンドの説明

show # 現在のコントローラーの状況を確認する。

devices # デバイスの一覧の表示

scan<on/off> # デバイスの更新設定。（ペアリングが終わったら scan offで無効に戻しておいた方が良い）

power <on/off> # コントローラーの電源のOn／Off

pairable <on/off> # ペアリングの有効／無効

discoverable <on/off> # Set controller discoverable mode

pair MAC\_ADDRESS # ペアリングを行う

trust MAC\_ADDRESS # 認証を行う。（初回接続ではConnected が yes になった後に直ぐに noになる）

block MAC\_ADDRESS # 特定デバイスのブロック。ブロックの解除はunblock MAC\_ADDRESS

#### ラズパイでブルートゥース接続

<https://qiita.com/oko1977/items/9f53f3b11a1b033219ea>

#### ALSA

Advanced Linux Sound Architecture

複数のサウンドカードのデバイスドライバを提供する。現在では編集，ミキシング機能を追加しようとしている。

設定ファイルの場所：/usr/share/alsa/alsa.conf

#### PulseAudio

上記のALSAは１つのアプリケーションのアクセスしかできない。PulseAudioが複数のアプリの音をまとめて，ALSAにアクセスしている。

# 音量を70%程度に

pacmd set-sink-volume @DEFAULT\_SINK@ 0xb400

トラブルシューティング

[https://wiki.archlinux.jp/index.php/PulseAudio#.E3.83.88.E3.83.A9.E3.83.96.E3.83.AB.E3.82.B7.E3.83.A5.E3.83.BC.E3.83.86.E3.82.A3.E3.83.B3.E3.82.B0](https://wiki.archlinux.jp/index.php/PulseAudio" \l ".E3.83.88.E3.83.A9.E3.83.96.E3.83.AB.E3.82.B7.E3.83.A5.E3.83.BC.E3.83.86.E3.82.A3.E3.83.B3.E3.82.B0)

#### SOX

sudo apt install sox # sox というアプリが便利だとか。

　rec test.wav　# 録音ができる

<https://qiita.com/propella/items/4699eda71cd742cba8d3>

aplay # 音声データ再生

mpg321 # mp3再生

amixer #

set Master 100% # 音量の変更

cset numid=3 1 # アナログ HDMI は numid=3 2

## raspi-config

### Interface Options

#### P3 Overlay File System

公式：<https://www.raspberrypi.com/documentation/computers/configuration.html>

## ハードウェア

### 電源オフ

#### pythonスクリプトを使う方法

但し、スクリプトの実行をroot権限で行う必要がある。

解決策

１. 実行権限を付ける

sudo chmod 755 shutdown.py

意味あるのか？

2. systemdで実行する

[ /etc/system/system/powerbutton.service]（例）

[Unit]

Description=Power Button Service

[Service]

Type=simple

WorkingDirectory=/home/pi/power

ExecStart=/usr/bin/python3 /home/pi/power/main.py

StandardOutput=file:/var/log/powerbutton.log

StandardError=file:/var/log/powerbutton\_error.log

User=root

Restart=on-failure

[Install]

WantedBy=multi-user.target

3. /etc/rc.localで実行する

但し近年のLinuxではrc.localの実行は非推奨

#### /boot/config.txtを使う方法

devicetree overlayというLinux Kernelに付いている機能を、更に簡単に使うことができる。

[ /boot/config.txt ]

dtoverlay=gpio-shutdown # 標準ではGPIO3が使われる

dtoverlay=gpio-shutdown,gpio\_pin=5 # GPIOピンの指定

dtoverlay=gpio-poweroff,gpiopin=18,active\_low=1 # 完了したらgpio18をオン

補足

・標準ではGPIO 3だが、I2Cで使うピンなので、I2Cを無効化する必要がある

・RaspberryPiOS内にも詳細情報あり。 /boot/overlays/README　但し、長い

### GPIO

https://pinout.xyz/pinout

## 通信設定

### シリアル通信

#### USB→シリアル

市販ケーブルなど買うのが手っ取り早い。

232C：秋月　通販コードM-08343 \980 ※エレコムUC-SGT1も普通に認識，通信できた。20.07.06

RS485：変換基板BOB-10124 ケーブル：USB-RS485-WE-1800-BT

ls -l /dev/serial/by-id/ # 有効デバイスの確認　　例) /ttyUSB0などと表示される

#### 直接GPIOピンを使用

6番(Ground)、10番(RXD)、 8番(TXD) として使う。

初期設定の場合，baudrate は115200、パリティ無し、ストップビット1bit らしい

232Cの場合は7Vに変換，だが変換基板はあまりない。usb変換する方が一般的らしい。

先にraspi-config でシリアル通信を有効化する必要がある。その後，port名を確認。

ls -l /dev/ttyS0 # 直接ピンを使う時のポート名　ttyS0などと表示される。

ls -l /dev/ttyAMA0 # 注意！こちらはUART0用で，初期設定ではBluetoothのモジュールになっている。

#### pythonを使う場合

[ python ]

import serial

ser = serial.Serial( # シリアル通信の設定

port='/dev/ttyS0', # 使用するポート名の設定

baudrate= 9600,

bytesize= serial.EIGHTBITS, # SEVENBITS =7; EIGHTBITS =8

stopbits= serial.STOPBITS\_ONE,

parity= serial.PARITY\_NONE) # なぜかNONE以外指定できない 19.07.10

ser.write(data) # 送信

cuというコマンドを使っても良いらしい。

sudo apt install cu

### I2C

#### 事前準備(bash)

##### インストール

sudo apt install i2c-tools # インストール（最初から入っている場合が多い）

python3 -m pip install smbus # python用

sudo raspi-config # Interfacing Options でI2C を有効にする。

通信速度の指定

[ /boot/config.txt ]

dtparam=i2c\_baudrate=50000 # i2cの通信baudrateを指定。再起動が必要？

##### 配線

GPIO2(3)：Data，GPIO3(5)：Clock

通常この２ピンを使用する。こちらがバス番号1番。

この２つのピンは1.8kΩのプルアップ固定になっていて，**他の用途での使用に向かない**為注意する。

GPIO0(27)：EEPROM Data，GPIO1(28)：EEPROM Clock

I2Cとしても使えるが，通常HATのEEPROM（メモリ）通信に使われる。バス番号0番？

##### 通信テスト

i2cdetect -y 1 # バス1番に接続されているデバイスを調べる

sudo i2cset -y 1 0x20 0xff 0x01 b

sudo i2cget -y 1 0x76 0xF4 b # 読込み　引数〔i2cアドレス　レジスタアドレス〕

（参考）<http://hogetan.net/note/memo/pi_i2ccmd.html>

#### 使い方

##### i2c-tools

i2cdetect

構文：i2cdetect [-y] *bus\_num*

オプション

-F *bus\_num* 全指定されたバスが行える機能についての情報を表示

-l I2Cのバスを列挙

-V バージョンの表示

sudo i2cdetect -y 1 # 有効なチップのアドレスを調べる。

i2cset

構文：i2cset [options] *bus\_num* *chip\_addr* *data\_addr* [value] ... [mode]

オプション

-f 強制

-y 対話で全てyes回答

mode

b バイトデータ

w ワードデータ(16bit)

c 値無しのバイト？？

sudo i2cset -y 1 0x76 # 書込み

sudo i2cget -y 1 0x76 # 読込み

i2cset

構文：i2cset [options] i2c\_bus chip\_addr data\_addr [value] ... [mode]

オプション（i2cset特有のもののみ抜粋）

-m MASK

mode

i I2Cブロックモード

（参考）<https://moba1.hatenablog.com/entry/2019/10/09/114145>

##### SMbus

###### C言語

smbusの資料（英語）：<https://smbus2.readthedocs.io/en/latest/index.html>

###### python (smbus)

インストール

python3 -m pip install smbus2 # python 用のモジュールをインストール。

python3 -m pip list | grep ‘smbus’ # インストール済みかどうか確認できる

from smbus2 import SMBus

bus\_number = 1 # バス番号は通常 1

addr = 0x76 # 通信相手を表すアドレス

bus = SMBus(bus\_number) # I2C通信を扱うモジュール

bus.write\_byte\_data(addr,0x88,123) # 1byteの書き込み (通信相手，レジスタアドレス，値)

rd\_dt = bus.read\_byte\_data(addr,0xF7, 1) # 1byteの読込み(通信相手，レジスタアドレス，[バイト数])

主要メソッドの抜粋

read\_byte\_data(i2c\_addr, register, force=None) **1byte**の読込みで，対象レジスタの指定ができる

read\_block\_data(i2c\_addr, register, force=None) **指定レジスタから32byte**の読込み？

read\_i2c\_block\_data(i2c\_addr, register, length, force=None) **バイト数付き**で複数byteの読込み。

write\_byte\_data (i2c\_addr, register, value, force=None) **1byte**の書込みで，対象レジスタの指定ができる

write\_block\_data (i2c\_addr, register, data, force=None) 複数byteの書込み

write\_i2c\_block\_data(i2c\_addr, register, data, force=None)　 複数byteの書込み

（参考資料；英語）<https://smbus2.readthedocs.io/en/latest/index.html>

writeメソッドの注意点

第2引数の名前がregisterとなっていて，ここにはレジスタアドレスを入れるのだと考えてしまうが，メソッドの引数と実際に出力されるバイトは以下のとおり。

bus.write\_block\_data(0x76, register=0x01, data=[0x02, 0x03])

# > 0x76 のチップに [0x01, 0x02, 0x03] という3バイトが出力される。

確かにI2C通信の際の第1バイトは書込み先のチップのレジスタアドレスである事が多いが，必ずしもそうでは無い。writeメソッドのこの特徴と，通信相手のチップの仕様をよく理解しないと通信できずにハマる事になる。

（参考資料；英語）<https://stackoverflow.com/questions/45324851/>

##### pigpio

##### WiringPi を使う場合

### SPI

#### 概要

SPI通信はRPi.GPIOなどでも通信できるが，spidevというモジュールを使ったほうが圧倒的に楽。

terminal:

sudo echo 44 > /dev/spidev0.0 # 直接送信する方法。あまり使わない。

【 重要 】spi通信の注意点

まず，SPIは通信相手（スレーブ）にデータを書込む時も，通信相手のデータを読出す時も，まずマスター側が「制御データ」を送る必要がある。その他は通信相手の仕様をとことん読む事。

（有志による資料；英語；見やすい）<https://www.sigmdel.ca/michel/ha/rpi/dnld/draft_spidev_doc.pdf>

#### 事前準備

##### インストール

spidevは最初から入っている場合が多いが，有効化が必要

terminal:

sudo raspi-config # Interfacing Options でSPI を有効にする。

SPIの有効化はboot\config.txtを編集しても良い

dtparam=spi=on # コメントアウトを外すだけ

確認方法

lsmod | grep spi # SPIが有効かどうかの確認

ls -la /dev/spi\* # 有効かどうかの確認はこれでも良い

##### 配線

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | pin# | GPIO | 通信相手 | 備考 |
| CE0 | 24 | 8 | CS(CSB) | /dev/spidev0.0 |
| CE1 | 26 | 7 | CS(CSB) | /dev/spidev0.1 |
| SCLK | 23 | 11 | SCLKまたはSCK |  |
| MOSI | 19 | 10 | MOSIまたはSDI |  |
| MISO | 21 | 9 | MISOまたはSDO |  |

（補足）Terminalでgpio readallを実行するとピン番号は確認できる。

##### 通信テスト

SPIにはi2cdetect のようなツールがない。以下のツールが似た感じ。

wget https://raw.githubusercontent.com/torvalds/linux/master/tools/spi/spidev\_test.c # DL

gcc -o spidev\_test spidev\_test.c # コンパイル

./spidev\_test -D /dev/spidev0.0 -v # 実行 -Dはデバイスの指定

./spidev\_test -p ‘\xF9\x00’ # 送信データの指定

[ spidev\_test.c ] ※実行時にパラメータ指定するのではなく，ソースコードを書き換えてしまう方法

static const char \*device= “/dev/spidev1.1”; // ⇒ spidev0.0 に変更

static uint32\_t speed = 500000; // 通信相手によっては変更が必要かも？

#### 使い方

##### python

インストール

terminal:

python3 -m pip install spidev # python用モジュール。最初から入っている事も多い

python

import spidev

spi = spidev.SpiDev() #

spi.open(0, 0) # ( 系統番号, チップ番号[CE] )

spi.max\_speed\_hz = 10000 # 必ず設定する事

// メインメソッド

ret = spi.xfer2([0xFC,0x00]) # 相手のバッファメモリに書込み（送信）

ret = spi.xfer([0xFC,0x00]) # xfer - Chip-selectはブロック間で解放する必用がある（？

spi.close() #

詳細

spi.bits\_per\_word =0 # 範囲8～16　ラズパイでは読取り専用　def.=8

spi.cshigh =0 # 0:LOWの時に通信　1:HIGHの時　def.=0

spi.max\_speed\_hz = 10000 # ラズパイでは32MHz以下なので設定必須。def.=125MHz

spi.mode =0 # 通信モードの変更　範囲0~3 　def.=0

spi.lsbfirst =0 # LSBが先か，MSBが先か 　def.=0

read\_bytes = spi.readbytes(n) # 先頭からn バイト読み込む。⇒ 何に使うの？

参考資料（英語）：<https://www.sigmdel.ca/michel/ha/rpi/dnld/draft_spidev_doc.pdf>

spi.xfer2の使い方（英語）：<https://pypi.org/project/spidev/>（あんま詳しく載ってない）

spi.xfer2公式Git（C言語）：<https://github.com/doceme/py-spidev>

～SPI　つづき

#### WiringPiを使う場合

##### C言語

#include <wiringPi.h>

#include <wiringPiSPI.h>

unsigned char spi\_buff[4]; // 通信に使うバッファ

int ret\_set = wiringPiSPISetup (0, 10000000));  // (ch, 通信速度[Hz])

int ret\_gpio = wiringPiSetupGpio(); // 必要なのか？ 2021-07-15

pinMode(22, OUTPUT); //22pinを出力に設定 24では？2021-07-15

digitalWrite(22, 1); // B接なのでHIGHが非選択状態

digitalWrite(SS\_PORT, 0); // SSをLOW出力にして通信開始

wiringPiSPIDataRW(SPI\_CHANNEL, spi\_buff, 4); // 送受信 (ch, byte[], byte数)

digitalWrite(22, 1); // SSをHIGHにして通信終了

（情報元）<https://qiita.com/seamew201/items/bbf3961aacad3333fcf0>

（WiringPi公式Git）<https://github.com/WiringPi/WiringPi/blob/master/wiringPi/wiringPiSPI.c>

linux/spi/spidev.h というのもあるようだが？

#### Rpi.GPIOを使う場合

#### SPIのプロトコル：具体例

例 １：ADコンバータ

シンプルなADコンバータであれば，基本的に読込みしかない。

例 ２：BME280

レジスタ値を読む時と書く時で送信データを変えている。

コントロールbyteにはアドレス番地を入れるがMSB（最上位bit）が読みと書きで異なるし，続くデータバイトも読みと書きで異なる。

|  |  |  |  |
| --- | --- | --- | --- |
|  | コントロールbyte | データbyte | 具体例 |
| 書き | msbを0 + アドレス  e.g. 0xF2 → 0x72 | 書き込み値を1byteで表現 | ctrl\_byte1, data\_byte1, …　（多数対で可能） |
| 読み | msbを1 + アドレス  e.g. 0xF7 → 0xF7 | 読みたいbyte数分ダミーbytesを送る | ctrl\_byte, dummy1, dummy2　（2byte読みたい場合） |

資料：

BME280の使い方（日本語）<http://trac.switch-science.com/wiki/BME280>

（参考；python）https://github.com/SWITCHSCIENCE/BME280

[参考；Python]の解説：<https://kagemomiji.hateblo.jp/entry/2016/08/14/210335>

#### 資料

・raspi-configの代わりに/boot/config.txtを設定しても良い。

・50MHzを超える速度でのspi通信はうまくいかない可能性が高いらしい

（参考）<https://yomon.hatenablog.com/entry/2018/09/18/164843>

・ラズパイには２系統のSPIがあり，２つ目の系統を使う場合には以下の方法でブルートゥースかシリアルコンソールの無効化設定が必要（らしい21.01.13）

ブルートゥースを無効化

/boot/config.txt （下２行を追記）

dtoverlay=pi3-miniuart-bt # ブルートゥースを無効化

enable\_uart=1 # UART を有効に（これは不要かも？）

シリアルコンソールを無効化

/boot/cmdline.txt

⇒ console=ttyAMA0,115200(またはconsole=serial0,115200) を削除

（参考）<https://yomon.hatenablog.com/entry/2018/09/18/164843>

転送モード

|  |  |  |  |
| --- | --- | --- | --- |
| mode | クロック極性  (CPOL) | クロック位相  (CPHA) | 備考 |
| 0(0b00) | 0:正論理 | 0:クロックが0→1の時 | mode0が標準。だいたいこれでOK |
| 1(0b01) | 0:正論理 | 1:クロックが1→0の時 | 正論理はアイドル時LOW，通信開始時HIGH |
| 2(0b10) | 1:負論理 | 0:クロックが0→1の時 | 負論理はアイドル時HIGH，通信開始時LOW |
| 3(0b11) | 1:負論理 | 1:クロックが1→0の時 | mode0か3の場合が多い。 |

（補足）

CPOL：Clock Polarity CPHA ：Clock Phase

modeについて（わかりやすい）：<http://dlnware.com/theory/SPI-Transfer-Modes>

ここでHIGHとかLOWとか言ってるのはSCLKの話。CS（Chip Select）は常にHIGH→LOWの時に通信開始。

## 管理

# Terminal

lsb\_release -a OSのバージョンを確認

uname -a カーネルの確認

getconf LONG\_BIT Linux のカーネルの bit数を確認

dpkg -l パッケージの確認；dpkg --serch nano など色々検索方法がある

lscpu cpuの情報 cat /proc/cpuinfo でより詳細な情報

df -h ディスクの使用状況の確認

nano /etc/passwd/ ユーザー一覧 （テキストエディタはnanoでなくても良い）

vcgencmd get\_throttled 電圧の状態を調べる

### オートログイン

autologin-user=pi

参考資料

<https://wiki.ubuntu.com/LightDM>

### 自動起動

#### 概要

パッケージ（アプリ）の自動起動は以下の方法がある。（偏見によるラズパイでの使用頻度順）

|  |  |  |
| --- | --- | --- |
| 方法 | 概要 | 詳細 |
| rc.local | OS起動時 | スクリプト。root権限で実行。 |
| crontab | 時間などで起動 | パッケージ。ユーザー権限で実行される。 |
| systemd | デーモン | .serviceファイルを作成する。  正規の方法ではあるが少し面倒だし，少し知識が要る。 |
| autostart |  | 詳細不明 |
| shのスクリプト | ログイン時 | スクリプト。複数の種類がある。  但し元々の趣旨に添わないので非推奨。 |

#### rc.local

Linux OS の起動時に自動的に root 権限で実行されるスクリプト。

#### crontab

crontab -e

@reboot # cronデーモンの再起動時

#### systemd

（参考）[自作サービス](#_自作サービス)

bashのスクリプト

基本的に環境変数（特にPATH）を設定する為のスクリプト。

利用できない事はないが，本来の趣旨に添わないので非推奨。

### host名の変更

デスクトップモードは

Raspberrypiの設定 →

コンソール

/etc/hostname #

/etc/hosts # こちらも変更しないと、再起動後に元に戻ってしまう？

### バックアップ

バックアップは以下の２通りの方法がある。

#### SD Card Copier（GUI）

 →　アクセサリ　→ SD Card Copier

#### ddを使用する場合（CUI）

SD Card Copierでのコピーは楽で良いが，バックアップデータにつき１枚のSDカードが必要。

簡単とは言えないが，ddでバックアップを取ると .img ファイルをPCで保管できるので便利。

なお，ddを使う為に別のLinux機（ラズパイやUbuntu；Virtual Boxでも可）を用意する必要がある。

補足：windowsではWin32DiskImager というアプリを使うと良いらしい

ここでは，SDカードのデバイス名はsdbとし，以下はUbuntuでの操作とする。（多分Raspberry Pi OSでも同じ）

前準備：

lsblk | grep “sd.” # デバイス名の確認，かつSDカードの情報を確認。

sudo umount /dev/sdb2 # 先にアンマウントして置く。（GUIで操作しても良い。）

ファイルシステムの縮小：

必須ではないが，これをやらないと書き込みに時間がかかってしまうし，保管先のストレージが圧迫される。

どちらにせよ同じ16GBのSDカードでも容量が若干異なる事があるので，やっておいた方が良い。

sudo e2fsck -p -f */dev/sda2* # 不整合や破断がないかどうか確認（resize2fsの為に必要）

sudo resize2fs -P */dev/sdb2* # 必要なファイルシステムの大きさを見積もる

> Estimated minimum size of the filesystem: **962363** 単位はブロック(4kb)

# 962363 [blocks] × 4 = 3951536[kb] ÷ 1024(kb → Mb) = 3858.92[Mb]

# 3858.92 + 100M（バッファ）= 3958.92

⇒ resize2fsでファイルシステムの大きさを変える時は繰り上げた値 **3959**M を使う。

sudo resize2fs -p */dev/sdb2* **3959M** # ファイルシステムのサイズを一時的に小さくする。

# resize2fs では実数（小数点）が使えない点に注意

～バックアップ：ddを使う場合　続き

fdiskでの操作：

partedで作業しても良い。やる事は同じ。パーティションテーブルがgptの場合はgdiskを使う。

sudo fdisk　/dev/sdb # fdisk の対話モードへ入る。　※sdb2ではない事に注意

Command (m for help): p # 詳細情報を表示：単位はブロック[=4kb] である点に注意。

>　Device Boot Start End Blocks Size Id System

>　/dev/sdb1 8192 96042 87851 42.9M c W95 FAT32 (LBA)

>　/dev/sdb2 **98304** 60751871 60653568 28.9G 83 Linux

Command (m for help): d → 2 # パーティション番号2を削除

Command (m for help): n　→ p → 2 # primary のパーティション２を再作成

※first sector は pコマンドで確認したStart の値を使う：この場合は98304

※last sector は 先ほどresize2fs で指定した値よりも大きめに：この場合 +4.2G　など

※ext4 signature は削除しない [N] （意味は分からないが20.07.13）

Command (m for help): p # サイズ変更を確認。また，最終セクタ数を控える。

>　/dev/sdb2 98304 **8896511** 8798208 4.2G 83 Linux

Command (m for help): w # 書き込み

sudo resize2fs /dev/sdb2 # ファイルシステムのサイズを再拡張。（サイズ指定が無いので最大まで拡張）

DDでバックアップ

# DDで用いるサイズ数の計算

#　まずはバイト数に直す。セクタサイズは通常512b なので (8896511+ 1) × 512 = 4555014144b

#　bs (block size) とcount を調整しつつ設定する。

#　めんどくさければbs=512にしてセクタサイズをそのまま使う。

#　バックアップを取る。（必要時間は約4min/1Gb）

# 補足：4555014144b ÷ 16777216[=16Mb] = 271.5 → 繰り上げで **272**を使う

sudo dd if=/dev/sdb of=./raspi-backup.img bs=16M　count=272 status=progress

書き込み（復旧）

ここでは，バックアップ先のSDカードのデバイス名はsdcとする。

# 書き込む

sudo dd if=./raspi-backup.img of=/dev/sdc bs=1M　status=progress # bsの指定は必須ではない

#　xzcat rspi.img.xz | sudo dd of=/dev/sdc # 普段はxzで圧縮しておく場合

# パーティション，ファイルシステムを拡張（元に戻す）

sudo fdisk　/dev/sdc # fdisk を使ってパーティションをリサイズ

Command (m for help):　p # 第２パーティションの**開始セクタを控え**ておく

Command (m for help): d → p → w # 上の例を参考。最終セクタまで拡張しておく。

sudo resize2fs /dev/sdc2 # ファイルシステムを再拡張。（サイズ指定が無いので最大まで拡張される）

参考）<https://qiita.com/narista/items/2812e6f596f956bc8e89>

#### DDのバックアップ：補足情報（まとまってない）

ファイルサイズの拡大

バックアップの為に一時的にパーティション２（sdb2など）のサイズが小さくなってしまっているので，

SDカードへのリストア後はファイルサイズを拡大した方が良い（必須ではない）

方法１：raspi-config → 7 Advanced Options → A1 Expand Filesystem

この方法が一番簡単で良い。

方法２：fdiskやGParted などを用いてフォーマットを変更する

GParted

パーティション変更アプリ。こちらの方がGUIで使いやすいのでインストールして置く。

sudo apt-get install gparted

# 蛇足情報：sd formatter でフォーマットしたSDカードはどうゆう訳か 4Mbくらいの余裕をもたせてある

resize2fsとfdiskを使ったパーティションリサイズ部分はこちらで対応できる（多分 20.07.13）

なお，fdsik, gdisk, parted, gparted 共通で言えるが，ディスクをアンマウントしない事にはパーティション変更ができない。resize2fsではアンマウントなしでパーティションやファイルサイズを変えられる。

小ネタ（まとまってない）

dd if=/dev/mmcblk0p1 of=/dev/sda bs=512 count=1 # MBRをコピー（パーティション情報も含まれる）

　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　# bs=446 でパーティション情報を残す事が可能

マウント／アンマウント

sudo blkid # デバイス名の確認はこれでもいいらしい

ls -l /dev/disk/by-uuid # これでも

df -h # マウントの確認。

umount /dev/*s**db1* # マウントされている場合はアンマウントする。

fdiskでの単位はセクタ(512b) なので計算直す。

<https://emc-craft.xyz/raspberrypi/backup_sdcard/>

<https://dev.classmethod.jp/articles/raspberrypi-backup-restore/>

Command (m for help): x # p コマンドで　255 heads, 63 sectors/track と表示されていなければエキスパートモードへ

Expert Command (m for help): h　→ 255 # ヘッド数を255に

Expert Command (m for help): s　→ 63 # セクタ数を63に

#シリンダ数を計算する

#cylinders = FLOOR ((SDカードのバイト数) / 255 / 63 / 512 )

この例では、 2021654528 / 255 / 63 / 512 = 245.79. なので 245 にする(切り捨て).

Expert Command (m for help): c　→　63 # 計算値を入力 defaultで計算値出てるけど？

Command (m for help): r # ノーマルモードに戻る

Command (m for help): n → p → 1 # ブート用のFAT32パーティションを作成

First sector: Enter(デフォルト) #

Last sector: +50 #

Command (m for help): a ( → 1 ) # 起動可能にする

Command (m for help): n → p → 2 # RFS(Root File System) 用のLinuxパーティションを作成

First sector: Enter(デフォルト) #

Last sector: Enter(デフォルト) #

#### リモート

こんな事ができるらしい。

$ ssh pi@192.168.10.11 “sudo dd if=/dev/mmcblk0 bs=1M | gzip -” | dd of=~/backupimage.gz

⇒ 但し、パーティションサイズの変更とかまではできない（はず）

### その他

#### SDカードの寿命を延ばす設定

<https://iot-plus.net/make/raspi/extend-sdcard-lifetime-5plus1/>

## bashコマンド

### 概要

ラズパイ固有の物や，関係の強いもののみをまとめる。

それ以外は，別ドキュメントLinux.docxを参考。

### raspi-config

対話式画面では無く、CLIコマンドとしても使用可能

sudo raspi-config nonint do\_hostname raspi4

公式：[https://www.raspberrypi.com/documentation/computers/configuration.html#raspi-config](https://www.raspberrypi.com/documentation/computers/configuration.html" \l "raspi-config)

### vcgencmd commands

CPUやGPU関連の情報を取得する

|  |  |
| --- | --- |
| RPI vcgencmd commands | 意味 |
| vcgencmd measure\_temp | CPU温度 |
| vcgencmd measure\_clock arm | CPU周波数 |
| vcgencmd measure\_voltsp | 電圧取得 |
| vcgencmd get\_mem arm | CPU(arm)メモリ使用量 |
| vcgencmd get\_mem gpu | GPU(gpu)メモリ使用量 |

公式：[https://www.raspberrypi.com/documentation/computers/os.html#vcgencmd](https://www.raspberrypi.com/documentation/computers/os.html" \l "vcgencmd)

## 補足情報

### Desktop

2016年頃にラズパイのデスクトップ環境はLXDEからPIXELへ変更された。

LXDE GTKツールキット（C言語）を利用。

PIXEL ラズパイ専用

ちなみに

LXQt Qtツールキット（C++）を利用

### 名前解決

ラズパイの名前解決の方法。

terminal（ラズパイ上）

hostname #ラズパイのホスト名の確認

> raspi01 だったとする

同ネットワーク内から

ping raspi01.local で応答が返ってくる。

http://raspi01.local:8080 8080番ポートでWebサーバーを起動している場合

## サービス

### 概要

#### サービス関連のコマンド

sudo systemctl start [サービス名] # 例 abc.service というファイルなら start abc

# (opp) stop　(see also) restart

sudo systemctl status [サービス名] # 現在の状態が分かる

sudo systemctl daemon-reload # スクリプトを変更したりするとdaemonの再起動が必要

sudo systemctl enable [サービス名] # 自動起動サービスとして有効化する (opp) disable

systemctl is-enabled [サービス名] # 自動起動認定されているか確認

systemctl list-unit-files -t service | grep [サービス名] # 起動設定の確認

systemctl -t service # <= 起動中サービス一覧の確認

systemctl -t service –all # <= 全サービス一覧の確認　状態も確認できるので便利

その他：

ログは /var/log/syslog に保管されている。

注意点

・スクリプト内部にshebang ；ファイルに実行権限 (x) をつける。

・スクリプトからファイルにアクセスする時は， ./ などとせずにフルパスを記述する。

（恐らく，実行時にカレントディレクトリが systemctl のになっている）

・スクリプトは /opt/ ディレクトリに入れた方が良い　と言う情報があるが理由は不明

### Raspberry Pi OS標準のサービスのまとめ

（補足）すべて後ろに .serviceが付く。（例）cron ⇒ cron.service

service –status-all で見れる

alsa-utils ：オーディオ。サウンドカードのデバイスドライバ。(ALSA)

avahi-daemon ：mDNS（ローカルDSN）を提供。

bluetooth ：文字通り。

cron ：定時間経過毎にタスクを実行(crontab) ユーザー権限で実行される。

dbus ：アプリケーション間でメッセージ送信

dhcpcd ：DHCPの提供。IPアドレスの固定。

dphys-swapfile ：スワップ領域（仮想メモリ範囲）を管理する。

fake-hwclock ：システムの終了時に現在時刻を記憶する。仮想RTC。

hostapd ：無線LAN(wifi) のアクセスポイントを提供する

kmod ：カーネルモジュールをロード、挿入、削除

lightdm ：ディスプレイマネージャ（デスクトップモードのみかも？ 19.11.05）

/etc/lightdm.conf

networking ：

nfs-common ：nfs（ネットワークを用いたファイル共有）。

paxctld ：PaX（メモリの最小権限アクセス）のコントロールデーモン。

plymouth ：splash screen（起動時に表示される小さな画面）の制御

procps ：プロセス監視を行う。

rng-tools ：真乱数と疑似乱数の間のブリッジとして動く

rpcbind ：

rsync ：高速なファイル同期を行う。

rsyslog ：ローカルおよびリモートサーバーのログ管理。

ssh ：文字通り。

triggerhappy ：ホットキー（ショートカットキー）の管理

udev ：デバイス管理ツール

x11-common ：X Window System（ウィンドウシステム）を管理（デスクトップモードのみ？）

#### dnsmasq

DHCPのアプリ

# 以下の設定値を書き込む　という情報もある

listen-address=192.168.1.10 # listen するアドレスを明示的に指定

bind-interfaces

server=8.8.8.8

domain-needed

bogus-priv

### 自作サービス

【2021年1月22日以降，Linux.docxの方をマスターとしこちらは簡略化する】

#### 概要

自動起動プログラム（サービス）を作成したい場合は /etc/systemd/system に .service ファイルを作成する。（Unitファイルと呼ぶ）iniファイルのような形式。

例えばmy\_serviced.service というファイルを作成すると，「my\_serviced」という名前のサービスが作成され，以下のように実行できる。

sudo systemctl start my\_serviced # 今，開始する

sudo systemctl enable my\_serviced # OS起動時にサービスが自動起動するようにする

#### 具体例

[Unit] # サービスの情報（メタデータ）

Description = Some Description # 　サービスの説明。空白文字も使える。

[Service] # メインの設定セクション

Type=simple # 　通常起動,，通常終了（デフォルト）

ExecStart= /opt/do\_something.sh # 　サービス起動時のコマンド

[Install] # サービスの有効化／無効化に関する設定

WantedBy = multi-user.target

（補足）

.timerファイルと併用すると，一定時間毎に動作するサービスが作成可能。（compare）cron

普通に/etc/systemd/system内にあるファイルを覗くと具体例が見られる。

具体例

参考資料

man 5 systemd.unit Linux標準のマニュアル。但し長文。

manと同じもの？（英語）<https://www.freedesktop.org/software/systemd/man/systemd.service.html>

### プログラム系

#### Python

##### Python の概要

特徴の箇条書き

・インタプリタ型 ：実行結果がすぐに分かるので学習に最適。

・豊富で強力なライブラリ ：短時間，最低限のコード記述で開発が可能。

・インデント型 ：コードが見やすく，メンテナンス性が良い。

・誰が書いてもほぼ同じコードになる ：このメリットは大きい。

・数行のシンプルな処理から大規模なパッケージ，機械制御からWebまで，幅広い開発が可能。

###### インタプリタ

通常，プログラムはコンピュータが理解できるように機械語（0と1のみの羅列）に “翻訳” する必要があり，この翻訳を**コンパイル**と言う。コンパイルされたファイルを「実行ファイル」，または「バイナリファイルと呼ぶ」。拡張子は Windows系では[ .exe ]。（Linux系では特に規定なし）

コンパイルを１冊の本の翻訳に例えるなら，インタプリタは会話の通訳で，プログラマが書いたコードを１行ずつコンパイルして実行する。それぞれにメリット／デメリットあるが，インタプリタ型は実行結果がすぐ分かるため，プログラム学習しやすい。プログラム初心者はまず，**Pythonはインタプリタ型言語**である，と言う事を理解しておく。

（補足）他のインタプリタ型言語はJavaScript, Rubyなど。

###### ライブラリ

　プログラム的な道具や機能の集まりの事。近年のプログラム言語では**よく使う機能などは初めから準備されている事が多い。**ライブラリを活用すればプログラム職人でなくとも簡単にアプリ開発ができる。

蛇足だが，以前はこのライブラリ部分も各個人や各企業で作成，管理されていた。

##### Pythonの使い方

###### コードエディタ

はじめの学習はラズパイでなくパソコンでやった方が良い。また，開発時もパソコンで作ったコードをラズパイにコピーするのが良い。パソコン向きコードエディタのおすすめはVisual Studio Code

Visual Studio Code：無料で使える強力なコードエディタ。C++, Java, PHP, Perl, Ruby, Powershell, html… と幅広い。

VSCodium ：Visual Studio Codeの100%オープンソース版。ほとんど変わらないらしい。ラズパイでも使える。

ラズパイでしか開発できない時は（gpioを使用したアプリとか），Raspberry Pi OSに最初から入っているコードエディタgeanyなどを使うと良い。ラズパイはVSCodiumも使えるが，多少不具合もある。（参考）[VSCode](#_VSCode)

（補足）Linux でのコードエディタはVim 一択　との意見をよく見るが，初心者にはまず無理だし，現代向きではないと思う。

###### コマンド

コマンドアプリでpythonを使用する事もできる。Windowsならコマンドプロンプト，Macではターミナル。(※１)

但しこの場合はPythonをインストールする必要があり，pathも通しておく必要がある。(※２)

Windowsでの使用例

# Command Prompt

C:\>python **::** python と打つとpython が起動する。

Python 3.6.x 　(...)

Type "help", "copyright", "credits" or "license" for more information.

>>>

最後の >>> が出るとPythonのコマンドが打てる。（逆にコマンドプロンプトのコマンドは打てなくなる）

コマンドを打つと結果が即座に反映される。この様な即時実行画面を「対話型」などと表現する。

なお，終了するには

>>> exit() # 最後の () が必要。 ※() は関数；functionである事を示す。

(※1)　 MacOS はUnix ベースで，Linuxシステムと非常に近いらしい。その為か、コマンドのアプリの名前がLinuxと同じ。

(※2)　インストールする時に 「Python3 をpathに追加する」 と言うのが出てくると思う。

なお、そこで追加し忘れてるとちょっと面倒。　⇒（参考）<https://www.sejuku.net/blog/54425>

###### サンプル

初心者向きのサンプルはWebに豊富にあるので，ここでは簡単な計算の例のみ載せておく。

まずは簡単な計算機として使ってみると良い。

# Python

arg = 5/3 # 変数 arg を作成して値を代入　算数っぽく書くと， 5 ÷ 3 = arg となる。

print(arg) # 変数arg の内容を表示

WebのPython入門

<https://tutorial.djangogirls.org/ja/python_introduction/>

<https://docs.python.org/ja/3/tutorial/> # 和訳が変なところもある。また，このチュートリアルはかなり深い。

##### 管理

###### バージョン

ラズパイにはpython2.x 系とpython3.x系がある

python -V　 # バージョンの確認（又はpython -version）

python3 -V　 # 3.xは基本的にpython3

which python # 格納ディレクトリが確認できる。

オプション

-h ：ヘルプの表示。 --help　-? でも可能

-m ：sys.path からモジュール名を探し，メインモジュールとして使用。

（参考）<https://docs.python.org/ja/3/using/cmdline.html>

###### ライブラリ管理(pip)

通常pythonのライブラリ管理はpipを用いる。

但し，ラズパイの場合は apt(-get) の方が対応しているパッケージの種類が多いとか（後述）

sudo apt install python-pip

sudo apt install python3-pip # python3のpip

pip list | grep ‘spidev’ # pip のパッケージを検索。

pip freeze # pip にインストールされたパッケージとバージョンの確認

###### ライブラリ管理(apt)

$ sudo apt-get install python3-numpy

$ sudo apt-get install python3-imaging

$ sudo apt-get install python3-pandas

$ sudo apt-get install python3-matplotlib

（参照）<https://karaage.hatenadiary.jp/entry/2018/10/05/073000>

###### その他

一度以下のコードエディタを試してみる。

Gedit，Vim，Atom

##### 関連パッケージ

sudo apt install python-virtualenv # 仮想環境

#### PHP

通常通りaptでインストールできる。

sudo apt install php # インストール

php -v # バージョンの表示

php -m # モジュールの表示

php -r "echo date('Y-m-d');" # 簡単なコードを試す　-r　オプションでコマンドの直接実行

sudo systemctl restart php # phpサービスの再起動

（補足）apacheやnginxのインストールは後でも先でも良いらしい？

設定ファイル：

/etc/php/[ver]/cli/php.ini

/etc/php/[ver]/apache2/php.ini apache向きphpの設定

#### Java

##### インストール

sudo apt install openjdk-11-jdk -y

## ディレクトリ

### ~（ホームディレクトリ）

#### .config/autostart

### /boot/

#### /boot/cmdline.txt

OSがブートされたときに、Kernelに渡されたパラメータ（？？？）

空白で区切って１行で記述される

|  |  |  |
| --- | --- | --- |
| 項目名 | 詳細 | 初期値 |
| console | console=ttyAMA0,115200 のものもあるとか。 | serial0,115200　と  tty1 |
| root |  | PARTUUID=3de68f8d-02 |
| rootfstype | ルートディレクトリのファイルシステム | ext4 |
| elevator |  | deadline |
| fsck.repair |  | y$ |
| ipv6.disable | =1でIPv6無効 | 記述無し |
| logo.nologo | 起動の際，Raspberryの画像を表示しない | 記述無し |
| quiet | 起動メッセージを表示しない | 記述無し |
|  | /usr/share/plymouth/themes/ に |  |

#### /boot/config.txt

ラズパイにはBIOSが無い。その代わりになるファイル。

# GPIO 系

gpio=21=ip,pu ???

# dtoverlay 系

dtoverlay=gpio-ir,gpio\_pin=17 # 赤外線通信を有効にする

dtoverlay=gpio-ir-tx,gpio\_pin=18 # 赤外線通信を有効にする

dtoverlay=pi3-disable-bt # Bluetoothを無効化。（要追加）

dtoverlay=gpio-shutdown # 電源オフボタン；標準ではgpio3

dtoverlay=gpio-shutdown,gpio\_pin=16 # GPIOピンの指定

dtoverlay=gpio-poweroff,gpiopin=26 # 完了したらgpio18をオン

# dtparam 系

dtparam=audio=on オーディオ（イアフォンジャック）を有効に

dtparam=spi=on SPI通信を有効にする

dtoverlay=spi1-1cs SPI1-1が選択される。（ラズパイには２系統SPIがある）

hdmi\_drive=2 意味不明（2021-12-06）

1行目がGPIO16にLowが入力されたらシャットダウンを開始する機能です。2行目がシャットダウン完了後にGPIO26にHighを出力する機能を有効にしています。GPIO番号は16, 17, 26, 27から選択できますが、両方同じ番号

# 開発環境

## C言語

### 基本

エディタはなんでもよい。Geany, leafpad，nano，vimなど好きな物を使えば良い。

パソコン上でVSCodeなどを使って，コピペかファイル転送が楽な気がする。

事前準備

terminal:

sudo apt install build-essential # ビルド・ツールをインストール

touch hello.c # 今からソースコードを書くファイルを新規作成。

サンプルソース

[ hello.c ]

#include <stdio.h> # printf を使う為に必要なツールを使います，という宣言

int main(int argc, char \*argv[]){ # エントリ・ポイント　プログラムがここから始まる，という「お約束」。

printf(“hello world\n”); # 最後の ; を忘れないように注意。　\n は改行の意。

}

コンパイル

terminal:

gcc *hello*.c -o *hello* # コンパイルする → 実行ファイルができあがる。

./hello # コンパイルした実行ファイルを起動。 ./ は現在ディレクトリの意味

-o オプションで出来上がる実行ファイル名を指定できる。

-o オプションを省略すると a.out という実行ファイルが作成される。

補足

・アプリケーションが大きい（複数のファイルを使う）場合にはmakefileやcmakeを使う方法もある

・標準ライブラリで無いものは -lオプションを使って，明示的にリンクの指定が必要。

### 複数ファイル

C言語で複数ファイルに分ける方法。

ファイル構成

/*multi\_files* ただの格納フォルダで，名前は何でも良い。

├ main.c エントリ・ポイントを含むファイルは慣例的にmain.c

├ hello.c 追加するソースファイル本体。

└ hello.h 追加するソースファイルのヘッダ。目録みたいな感じ。

ソースコード

[ main.c ]

// mainの方では stdio.h のincludeが必要ない。（むしろしない方が良い）

#include "hello.h" // <>　でなく，""　である点に注意。

int main(int argc, char \*argv[]){

sayHello();

}

[ hello.h ]

/\* ここにメソッドの説明書きがある事が多い \*/

int sayHello(); // メソッドの中身を書かない。

[ hello.c ]

#include "hello.h" // 真っ先に includeするべき　という声が多い

#include <stdio.h> // printf

int sayHello(){

printf("hello world\n");

}

コンパイル

gcc -o *hello* *main*.c *hello*.c # 必要ファイルをすべて記述。（.cファイルの名前だけでよい）

./hello # コンパイルした実行ファイルを起動。

複数ファイル時の注意点

・別ファイルでも使う変数にはexternをつける。

個人的にはexternは極力使うべきでないと考える。面倒でも，引数や戻り値などで渡すべき。

・複数ファイルで同じ名前の変数やメソッドを宣言しない。（スコープが異なれば宣言できる）

### Lチカ

#include <stdio.h>

#include <stdlib.h>

#include <wiringPi.h>

#define LED0 23

int main (void){

wiringPiSetupGpio();

pinMode(LED0, OUTPUT);

for(;;){

digitalWrite(LED0,HIGH);

delay(500);

digitalWrite(LED0, LOW);

delay(500);

}

return EXIT\_SUCCESS;

}

### 資料

※詳細は別ドキュメントC/C++かLinuxを参照

## C++

C++を使う時にはg++というコマンドがある。

g++は最初から入っている事が多い

g++ hello.cpp # C++ファイルのコンパイル。

## Visual Studio(Win) を使ってのネイティブ開発

概要

・開発言語はC++

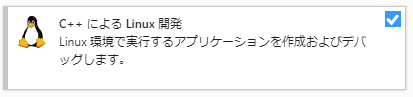
・CUIアプリしか開発できない。

・コンパイルされた実行ファイルはSSHを使ってリモート先に保存される。home/user\_name/project

・リモートデバッグも可能

・.out ファイルはLinuxでの実行ファイル。

手順

Visual Studio InstallerでC++によるLinux開発にチェックを入れる。

## Visual Studio Code でリモート開発

別ドキュメント　ソフトウェア情報：遠隔編集　に移動

## ライブラリ(C/C++)

ラズパイに特有のものだけここに記載する

なお，通常C言語用ヘッダーファイルは /usr/include/ に格納されている。

### <wiringPi.h>

（公式情報？；英語）<http://wiringpi.com/reference/>

wiringPiSetup() ;

wiringPiSetupGpio()

digitalWrite

構文：

### <pigpiod\_if2.h>

（公式情報？；英語）<http://abyz.me.uk/rpi/pigpio/index.html>

### <pthread.h>

### 資料

ラズパイの割り込みについて：<https://qiita.com/toshinaga/items/e65004cc207a46d17b7b>

## java

sudo apt install default-jre # 実行環境におけるデフォルトがインストールされるらしい

sudo apt install openjdk-11k # バージョンを指定する場合

java --version # インストールが成功したかどうか確認

# アプリ

ラズパイ特有のものでない限り，Ubuntuの方にとりあえずまとめる事にする 20.09.04

## 基本アプリ

### sh

Linux系のコマンドを実行する為のプログラム。間接的に使う事が多い。Win10やMacOS X にも入っている。

（補足）

このshは実際には/bin/bashへのシンボリックリンクである事が多い。

但し，ラズパイではbashでなく，**dash**を使っている。

ただ，これによってどう変わってくるのかは良く分からない 2021-01-25

### Terminal

テキストファイルへの保存

> を使うと，処理結果をテキストファイルに保存できる。　（　>> だと追記になる。）

例）

echo hello > /etc/home/pi/test.txt

以下の様にすると，複数行記述できるらしい 19.09.23

nano > /etc/hostapd/hostapd.conf <<EOF

…

EOF

#### スクリプトファイル

Terminalでのコマンドをまとめて記載したテキストファイルを作成して，一気にまとめて実行をすることもできる。Windowsの.bat(.cmd) ファイルと同じ。拡張子は .sh

関数

関数を作る事も可能

#### その他

・sudo su super user に完全に切り替えて実行できる。sudo -s でも可？

・各コマンドのオプションは，ハイフン１つ型(-h)とハイフン２つ型 (--help) がある。 （効果は同じ）

（補足）ハイフン１つ型が伝統的に使われていたが，分かりやすくする為にハイフン２つ型が後出で追加されたらしい

・なんらかのコマンド > text.text で結果をテキストファイルに保存できる。

#### 設定ファイル

.bashrc

#### 常用コマンド（抜粋）

ls # list segments の意。ファイルやディレクトリの情報を表示する。

-l [path] # 指定path内のファイルをリスト形式で詳細表示する。

### テキストエディタ

nano Console(CUI) 形式でのテキスト編集。

leafpad Desktop(GUI) 形式でのテキスト編集。

vi 中級者より上の人向けのテキスト編集。（サーバーを借りたりするとviしか無かったりする）

### nano

コマンドラインで動くテキストエディタ

#### 基本的な使い方

切り取り 範囲を選んでからCtrl+K → 選択切り取り ；範囲選ばずにCtrl+K → 一行切り取り

ペースト Ctrl + U

行頭への移動 Ctrl + A 行末への移動 Ctrl + E

次ページへの移動 Ctrl + Y 前ページへの移動 Ctrl + N

文頭への移動 Alt + \ 文末への移動　 Alt + /

任意の行への移動 Esc + G

後方検索 Ctrl + W 前方検索 Ctrl + Q

前方へ続けて検索 Alt + W 後方へ続けて検索 Alt + Q

置換 Ctrl + WR

編集の終了、保存

Ctrl + X で編集を終了 → Y 保存；N 保存しない；C　終了のキャンセル

#### nano起動時のオプション（抜粋）

-v # 閲覧モード

-m # マウスを使用可

-B # バックアップファイルの作成

-F # 新しいバッファーでファイルを読む

設定ファイル　$HOME/.nanorc

set autoindent #　インテンドを揃える

set backup #　[file name]~でバックアップを保存

nanoの公式doc（英語）：<https://www.nano-editor.org/docs.php>

## 基本操作用アプリ

### Linux標準

tar アーカイブを作成する。

アーカイブとは圧縮の無いzipファイルのような物。tarを使ってアーカイブ → 圧縮も可能。

cat ファイルの中身をテキスト表示する。表示だけで編集しない場合に用いる。

### Office

Office ：LibreOffice

$ sudo apt install libreoffice # 英語版？

$ sudo apt install libreoffice-l10-ja # 日本語版

$ sudo apt install libreoffice-help-ja #

### シリアル通信

・pyserial

シリアル通信用のパッケージ。Raspberry Pi OS Stretch には最初から入っているらしいが，一応確認しておく。

sudo apt search pyserial # pyserial を検索

sudo apt search --names-only .serial # 正規表現して，範囲を広げた検索。

sudo pip install pyserial # ない場合はインストール

・raspi-config

5. Interfacing Options：legin shellはDisable(No)；hardwareはEnable(Yes)

⇒ 再起動が必要。※

#Terminal

ls /dev/se\* #（又は　ls /dev/\*） /dev/serial0 があるか確認する。

ls -l /dev/serial0 # ⇒ ttyS0

sudo stty -F /dev/ttyS0 19200 # ボーレートの変更

設定ファイルの変更が必要。※デフォルトではシリアル通信がブルートゥースに設定されている為

# /boot/config.txt （下２行を追記）

enable\_uart=1 # UART enable

dtoverlay=pi3-miniuart-bt # BT disable

参考）<https://kondo-robot.com/faq/raspberrypi_wire_rcb4>

terminal でも通信は可能

gpio -g write 18 1 # RTS をon (18番にRTSを配線している場合)

echo “hello” > /dev/serial0

pyserial オンライン書類) <https://pyserial.readthedocs.io/en/latest/pyserial_api.html>

・RasPiComm　シリアル通信を行う時のライブラリらしいが？？？

## アプリ開発

### Geany

概要：Raspberry Pi OSに最初から入っていて，Thonnyよりは使いやすい。

注意点：

Geanyではshebangは無視され，常にPython2で動く。

Python3で動かすには：メニューのビルド → ビルドコマンドの設定→

で以下の２つをpython から python3に変更。

Compile：**python** -m py\_compile “%f” ⇒ **python3** -m

Execute：**python** “%f” ⇒ **python3** “%f”

（参考）shebang 実行時にインタプリタ（≒実行アプリ）を指定する。（前述の通り，Geanyでは無意味）

#! /usr/bin/env python3

### VSCode

※VS Codeはクライアント機（パソコン）上でVS Codeを動かして，VSCode上でSSHやった方が圧倒的に楽。

・1.32.0 にはバグがある。1.29.0 をダウンロードする

<https://qiita.com/kent77/items/2f1715c29f8fc43d26fb>

# 先にsuper userにしておく

sudo -s

# . <( wget -O - https://code.headmelted.com/installers/apt.sh ) # ダウンロード

sudo apt install code-oss=1.29.0-1539702286 # ver1.29.0 を指定

sudo apt-mark hold code-oss # アップグレードを控える

# 失敗したらこれを打ってみる

sudo apt install code-oss -y --allow-unauthenticated

# GPUがトラブルを起こす事もあるらしい

code-oss –disable-gpu # GPUを無効にして起動してみる

extention でpythonとpython doc は入れておく　19.07.11

### Scratch

公式サイト：<https://scratch.mit.edu/>

#### インストール

sudo apt install -y scratch3

2021-10-05現在、scratch1,2 はもうすでに使える状態ではない。scratch3のみが有効。

プロジェクトを保存すると .sb3 という拡張子のファイルが作成される。



#### ネコを動かす

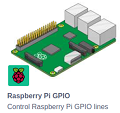
基本的に「1ブロック」が1つの実行単位となっているらしい。（右図参照）プログラムの実行中はブロックの境界線が黄色になり，

 の緑の旗が選択状態になる。赤丸を押すと実行停止。

処理の種類によってブロックの形が異なるらしい

|  |  |  |
| --- | --- | --- |
| 形 | 区分 | 意味 |
|  | 処理 |  |
|  | イベント | ボタン入力や何かしらの条件が変わった瞬間をトリガとする。1つのイベントに対し，右図のようなブロックの塊を作ることになる。 |
|  | データ処理 | 変数定義，代入，演算， |
|  | 条件 | 右図のようなブロックにはまる |

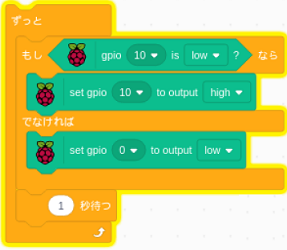
#### GPIOを使う

 拡張機能を追加 → Raspberry Pi GPIO（右図参照）

→ 左サイドバーの中にRaspberryPi GPIOが表示される。

Lチカの例

単純な無限ループ版 条件判断付き



### GitHub

概要

make

./configure # ダウンロードしたフォルダにあるファイル。実行するとファイル構成が正しいかチェックできる。

# ※コマンド名ではなく、ファイル名。

make # ソースファイルのコンパイル

make install # コンパイルしたアプリをインストール

GitHubを使用する前には、GitHubのサーバーに公開鍵を登録しておく必要がある。

公開鍵の作成方法はSSHの時と同じでssh-keygen を使う。

# git が無ければ sudo apt install git

git clone https://github.com/aaa/xxx.git [target\_path] # target\_path省略でホームディレクトリに

# ユーザー設定

git config --global user.name [名前] # これらが不要かも

git config --global user.email [メールアドレス] # これらが不要かも

# 公開鍵認証

ssh-keygen -t rsa -C "メールアドレス" # ~/.ssh/id\_rsa が作成される

# ssh-keygen -t rsa -b 4096 -C "メールアドレス" -Cと言う情報もある

# Enter file in which… ()内のパスがSSH keyの保存先となる。問題なければそのままEnter

xclip -selection c [target\_path] # 公開鍵の内容をクリップボードにコピー xclip は要インストール

# GitHub で公開鍵を登録

# 1.GitHubにログインし，右上のプロフィールをクリック -> Settingsにて設定画面に遷移

# 2.左のメニューからSSH keysをクリック

# 3. Add SSH Keyをクリック

# 4. Titleに識別できる文字列を入力

# 5. Bodyに先ほどクリップボードに入れた値をペースト

# 6. Add Keyをクリック

# 秘密鍵のパーミションを変更

chmod 600 (秘密鍵のファイル名)

# /.ssh/config ファイルの作成

# 以下内容

Host github.com

HostName github.com

IdentityFile ~/.ssh/id\_rsa # ここで指定するのは秘密鍵の方 .pub の付いていない方のファイル

User git

# ssh を用いて接続できればOK

<https://qiita.com/jun6231jp/items/5596ef0c912c7f57ef0a> Webアプリの情報

### そのほか

Scratch

2021.09.27現在，Scratch2以前のものは使えない。

BlueJ

Javaの開発環境らしい。

Phython ：Python IDLE

## Web

### memo

海外IPを弾く

ラズパイでのWebアプリの作成方法

### avahi

（アバヒ）マルチキャストDNS(mDSN) を提供する。

sudo vi /etc/avahi/services/ssh.service # SSHサービスを登録

nginx # 軽量な Webサーバー

sudo apt install nginx

### Apache

#### インストール

sudo apt install apache2

sudo apt install apache2-dev # 拡張機能もインストール

netstat –listen # これでも動いているか確認できる

# Raspberry Pi OS は設定ファイルにapache2.conf を使う　（参照）[設定ファイル](#_設定ファイル)

# Red Hat系では /etc/httpd/conf/httpd.conf を編集する

sudo service apache2 start # 起動

sudo service apache2 stop # 停止

sudo service apache2 restart # 再起動

apachectl -v # apache のversion 確認

ブラウザを起動してhttp://localhost　に接続できたら成功。

/etc/apache2/ ディレクトリに設定ファイルが作成される。

/var/www/html にindex.html はある。

このindex.htmlを書き換えれば，Webページの編集ができるが注意点が２つ。

/var/www/html/ に［o+x：その他ユーザの実行権限］が必要

/var/www/html/index.html に［o+r：その他ユーザの読み取り権限］が必要

#### 設定ファイル

|  |  |
| --- | --- |
| 設定ディレクトリ | 設定内容 |
| /etc/apache2/apache2.conf  （環境によって  ファイル名が違うらしい） | メインの設定ファイル。Apacheの起動時に読み込まれる。  httpd.conf：CentOs などの場合  apache2.conf ：Debianなどの場合  ・ディレクトリごとの許可設定<Directory> |
| /etc/apache2/site-available | 個別のサイト設定を 000-default.confをもとに新規作成  ドキュメントルートの変更 e.g. /var/www/html |
| /etc/apache2/conf-available | 必要に応じてsecurity.confなどを修正  security.conf |
| /etc/apache2/mods-available | モジュール設定はここ |
| /etc/apache2/port.conf | port変更の設定はここ? ``(Listen 8080など) |
|  |  |

available は選択可能。enabled は現在有効になっているもの。

.htaccess

apacheで使われるhtmlサーバーの分散設定ファイル。root権限が無くても設定可能。

htmlサーバーへのアクセス毎に読み込まれる。

<https://qiita.com/sunnyG/items/7e5bd6e8dc9b04c9978e>

#### mod\_wsgi

sudo pip3 install mod\_wsgi # インストール

# python2/3 分かれている場合はpipでなくpip3 とする（apt でもいいらしい？19.07.26）

pip3 show mod-wsgi # インストールの確認

sudo apt install libapache2-mod-wsgi # Apache2 でwsgi を使う為のライブラリ

mod\_wsgi-express start-server # 確認用の簡易WSGIサーバー（Apcheと無関係）を起動

# http://localhost:8000及びhttp://localhost:8000/cgi-binでアクセスできる　（teminalに戻りCtrl+Cで終了）

sudo mod\_wsgi-express install-module # mod\_wsgiのモジュールをインストール

# モジュールのパスが表示されるので，念のためファイルマネージャで確認しておく

# モジュールの読み込み設定をする

# /etc/apache2/mods-available/wsgi.load の一行目を

# LoadModule wsgi\_module /usr/lib/apache2/modules/mod\_wsgi[確認したモジュール名] とする

a2enmod wsgi # モジュールの有効化

sudo a2enmod cgi # ApacheでCGIモジュールを使えるようにする

# /var/www/cgi-bin というディレクトリを作る　※公開ディレクトリ （html下）にはしない事！ chmod 775しておく

# /etc/apache2/conf-available/serve-cgi-bin.conf でCGIを動作するディレクトリを指定

#ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/

ScriptAlias /cgi-bin/ /var/www/html/cgi-bin/

#<Directory "/usr/lib/cgi-bin">

<Directory "/var/www/html/cgi-bin">

a2enconf serve-cgi-bin.conf # 変更を適用

※注意点として，.py ファイルの権限が755であり，かつ.pyファイルの１行目に

#!/usr/bin/env python

を書く必要がある。これはshebangと呼ばれている。sharp bang の略らしい。

また，windowsでスクリプトを書くとlinuxと改行コードが異なり上手く行かない。

<https://qiita.com/lamplus/items/9877849d3108e2c6d0df> # 主な情報元：

<https://docs.python.org/ja/2.7/howto/webservers.html> # python公式：

<https://modwsgi.readthedocs.io/en/master/> # wsgi公式：

<https://qiita.com/YasuyukiKawai/items/231821dd22a72194b3fb>　# CGI利用

<https://tech-blog.monotaro.com/entry/2018/07/04/084733>　複数のwsgi

#### エイリアス設定

/etc/apache2/sites-available/000-default.conf を編集

WSGIScriptAlias /test\_wsgi /var/www/html/test\_wsgi.py

などとする？？？

Alias リクエストのURLパス 実際のディレクトリ

#### その他情報

エラーログに Forbidden が出た場合

<https://engineers.weddingpark.co.jp/?p=1031>

ユーザーとグループ：

/etc/apache2/apache2.conf (≒ httpd.conf )に書いてある。

デフォルトでは：

User apache

Group apache

となっている為，ラズパイに登録されたユーザー，グループでない為Other権限になってしまう。

ユーザー名

apacheはLinux上ではwww-data というユーザー名を使う。

cat /etc/passwd | grep www

> www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin

sudo gpasswd -a www-data i2c # i2c通信が必要な場合

sudo gpasswd -a www-data gpio # GPIO を制御したい場合

ログ

/var/log/apache2 にログがある。

### nginx

#### PHPを使う

<https://qiita.com/Brutus/items/27525deedb0eea1b35b8>

### dhcpcd

概要：DHCPを提供。IPアドレスの固定。

#### allowinterfaces

指定したインターフェイス名をカーネルに渡して有効にする (opp) denyinterfaces

例）

denyinterfaces wlan\* eth0

資料：

<https://wiki.archlinux.jp/index.php/Dhcpcd>

<https://www.daemon-systems.org/man/dhcpcd.conf.5.html> (英語)

### hostapd

概要

ソフトウェアアクセスポイントを提供

nl80211対応の無線デバイスが必要。

iw list # iw無線LANネットワークの設定を行うコマンド

### iptables

#### 概要

パケットの宛先管理（インターネット層，トランスポート層）を行うアプリ。パケットフィルタリング型ファイアウォール。

（補足）ファイアウォールにはゲートウェイ型も存在する。

（※iptablesは比較的難しく，UFW；Uncomplicated Firewallというファイアウォールはもっと容易に扱えるらしい20.09.03）

matchするパケットをtargetの処理とする，と言った形で設定。ルールは上から順番に優先適応される。

**「全ての通信を遮断した上で必要な通信だけ許可する」**がセキュリティ上の基本

また、iptablesはTerminalで１つずつコマンドを打って使用するのが基本。

⇒ .shファイルを作成しておいてまとめて設定した方が間違いも少なく、管理も楽。

#### リスト表示

sudo iptables　 -L # リストの表示。

# 結果

(policy ACCEPT) 基本ポリシー。どのルールにも当てはまらない場合、ここで指定した処理を行う。

target 処理の指示。以下に示すtargetテーブルの内から１つ以上選んで指定する。

prop プロトコル。allは文字通りすべて。

opt オプション

source パケットの送信元。anywhere はすべてのノードを指す。

destination パケットの宛先。

残り 備考情報が表示される

sudo iptables -L -v -t filter # -v より詳細なリストの表示。 -t テーブルの指定。

sudo iptables -t nat -n -L # オプションの順番は決まっていない？

-n オプション DNSの逆引きを避ける

-z オプション パケットとバイトカウンタを零に

-v オプション より詳細なリスト表示

（参考）[オプション](#_オプション)

#### 書式

基本

iptables 〈テーブル〉〈コマンド〉 〈マッチ〉〈ターゲット〉

書式の参考例：

iptables [-t *table*] {-A|-C|-D} *chain rule-specification*

ip6tables [-t *table*] {-A|-C|-D} *chain rule-specification*

rule-specification = [matches...] [target]

match = -m *matchname* [per-match-options]

target = -j *targetname* [per-target-options]

コマンド抜粋：

-A, --append 指定Chainに1つ以上の新しいルールを追加

-D, --delete 指定Chainから1つ以上のルールを削除

-C, --check chain 指定したルールにマッチするルールが指定されたチェインにあるかを確認する

（参考）[コマンド](#_コマンド)

（具体例）

iptables -t filter -A INPUT -i lo -j ACCEPT # ⇒ インターフェイス名lo からの入力を許可するルールを追加する

この場合

*table*名がfilter　（これは省略可能。-t の部分を省略するとtableはfilterとなる）

*chain*はINPUT。また、パラメータに -i lo を指定　⇒ インターフェイスが local loopback（自分自身） の時

*rule-specification* は -j ACCEPTとなる。ACCEPTはtargetテーブル。

#### テーブル

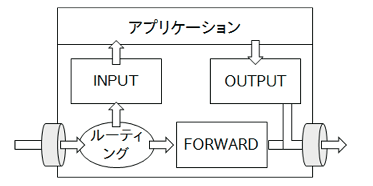
##### 概要

iptables には４つのテーブルがある。

filterテーブル（対象）

特にこの３つをChain と呼ぶ事がある。-tオプション省略の際のデフォルトtable。

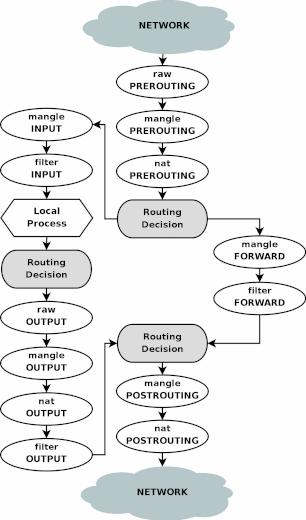
ローカルホスト宛の進入パケットは、インターフェースや入ってきた方向に関わらず、必ずこのチェーンを通る。

INPUT 自分(localhost)向きパケット（基本  
DROP）OUTPUT 内側で生まれて外に向かうパケット

FORWARD 転送（仲介ルーター以外の時はDROP）

natテーブル（アドレス変換）

PREROUTING 受信直後に宛先アドレスを変換する

POSTROUTING 送信の直前に送信元アドレスを変換する

DNAT 自身の使っているパブリック IP 宛へのアクセスを他のhostに宛てなおす。

SNAT DMZなどのローカルアドレスを隠すために、ネットワークアドレスを使用する。またはプライベートアドレスをパブリックアドレスに変換する。

MASQUERADE

SNATに似ているが、パケットが来るたびにIPアドレスを調べる。

その為に動的に DHCP IPアドレスを割り当ててくる PPP や PPPoE、 SLIP下でも動作する。

REDIRECT ？

##### 詳細

Mangle

mangle のみで使用可能なtarget

TOS(Type Of Service)

TTL(Time To Live)

MARK パケットに特別なマークを付ける

SECMARK セキュリティコンテキストを扱える機構でマークとして使う。

CONNSECMARK 単一のパケットのセキュリティコンテキストを全体にコピーしたりする。

概要

nat

ネットワークアドレス変換に用いる。ストリームの先頭パケットだけが一度だけこのテーブルを通る。

残りは先頭パケットに追従するのみ。その為、ここではいかなるフィルタリングも行うべきではない。

mangle

パケットの改変を行う。例えばTTL, TOS, MARK など。コネクション追跡の発動する手前でパケットを処理し、追跡で扱われないよう特定のパケットにマークを付ける。

filter

パケットフィルタリングを行う。iptableの核。DROP, LOG, ACCEPT, REJECT も問題なく行える。

raw

どのテーブルよりも先に適用される。NOTRACKターゲットを利用する為に考案されたテーブル。

実は比較的新しく、2.6後期のカーネルで実装。

適用されるビルトインChain

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | raw | mangle | nat | filter |
| INPUT |  | 〇 |  | 〇 |
| OUTPUT | 〇 | 〇 | 〇 | 〇 |
| FORWARD |  | 〇 |  | 〇 |
| PREROUTING | 〇 | 〇 | 〇 |  |
| POSTROUTING |  | 〇 | 〇 |  |

ここに詳細を書く

#### ポリシー(Policy)

そのテーブルの規定動作の事。厳密には、Chain内のいずれのルールにも一致せず、終点まで行ったパケットに適用される動作と言った方が正しい。

また、ユーザー定義Chain にPolicyを設定する事はできない。この場合に終点まで行ったパケットは、そのユーザー定義Chainの呼び出し元のChainに戻される。

指定したもの以外はすべてdrop（破棄） する

・安全性が高い

・利用可能なところに持っていくだけでもかなりの手間がかかる

指定したもののみdrop（破棄）する

#### チェーン(Chain)

ルールの集合体。

#### ターゲット

targetsテーブル（処理の指示）　以下のものはbuilt-in chains（削除できない）

ACCEPT　許可：パケットをすべて通過許可

DROP 破棄：パケットを破棄

REJECT拒否：パケットを拒否し，ICMP（制御）メッセージを返信

LOG：パケットのログを記録

　　--log-level どのログレベルを使うかを指示

　　　（例）iptables -A FORWARD -p tcp -j LOG --log-level debug

　　--log-prefix ログメッセージに特定のプリフィックスを付ける

QUEUE：

RETURN：

-j, --jump *target*

条件に合ったときのアクション（target）を指定

-g, --goto *chain*

　ユーザー定義チェインで処理を継続する。

　-jと異なり、-j でこのチェインを呼び出したチェインで処理が継続される。

　（よくわからない 19.11.09）

#### コマンド

-A, --append *chain rule-specification*

*指定chainの最後尾にルールを追加する。*

-C, --check chain *rule-specification*

指定したルールに一致したルールがあるか確認する。

-D, --delete *chain rule-specification*；-D, --delete *chain rulenum*

*指定*chainの指定ルールを削除する。

-I, --insert *chain* [*rulenum*] *rule-specification*

*指定chainにルール番号を指定してルールを挿入する。*

-R, --replace *chain rulenum* *rule-specification*

*指定chainの指定ルールを変更する。*

-L, --list [*chain*]-D, --delete *chain rulenum*

ルールの一覧を表示する。

-S, --list-rules [*chain*] ルールの一覧を表示する。-Lとの違いは？

-F, --flush [*chain*] 指定のchainを消去する。指定がなければ全削除。

-Z, --zero [*chain* [*rulenum*]] 指定Chainのすべてのパケットとバイトカウンタを零にする。

-N, --new-chain *chain* ユーザー定義chainの新規作成。同じ名前のターゲットが存在していてはならない。

-X, --delete-chain [*chain*] 指定したユーザー定義チェインを削除。引数無しですべてchainを削除。（built-in以外）

-P, --policy *chain target チェインのポリシーを指定したターゲットに設定*

-E, --rename-chain *old-chain* *new-chain* 指定chainの名前の変更（のみ）

-h ヘルプ

rule-specification = [matches...] [target]

match = -m *matchname* [per-match-options]

target = -j *targetname* [per-target-options]

#### マッチ

IPフィルタリングでmatchは２つの意味がある。

・あるルールの中の単一の条件倫理に一致する

・１つのルール丸ごとで１つのmatchと捉える場合。

##### 一般的なマッチ

情報元によっては、「パラメータ」のカテゴリに入っていたりする。

[!] -p, --protocol *protocol*

プロトコル( tcp，udp，udplite，icmp，esp，ah，sctp，all) を指定

[!] -s, --source *address[/mask][,…*]

　パケットの送信元のアドレス。IPアドレスorホスト名を記述

　（例）iptables -A INPUT -s 192.168.1.1/24

　（例）-s 192.168.1.1/255.255.255.0 #　/24 と同じ結果になる

　（例）-s 192.168.0.0/24 # 特定のネットワークを指定

　（例）-s !192.168.0.0/24 # 特定のネットワークを指定

[!] -d, --destination *address*[*/mask*][,*…*]

パケットの宛先のアドレス。IPアドレスorホスト名を記述

[!] -f, --fragment IPv4のフラグ（２番目，３番目）のパケットを参照する

##### 暗黙的なマッチ(implicit match)

TCPマッチ、 UDPマッチ、 ICMPマッチ が存在する。

tcpマッチ：-p tcp の時のみのmatch

--sport, --source-port ：（例）iptables -A INPUT -p tcp --sport 22

（例）iptables -A INPUT -p tcp --sport 22

　22:80 ：ポート22~80までを指す

　!22 ：ポート22番以外を指す

--dport, --destination-port

--tcp-flags

iptables -p tcp --tcp-flags SYN,FIN,ACK SYN # （例）

第1引数に評価されるTCPフラグを指定，第2引数に設定されていなければならないフラグを指定

指定可能なフラグは，SYN，ACK，FIN，RST，URG，PSH，ALL，NONE

--syn

--tcp-flags SYN,RST,ACK SYN の省略形。

##### 明示的マッチ(explicit match)

-m または --matchではっきりとロードする必要があるmatch

特定の通信を検査する拡張モジュールを指定。

先頭から末尾へ向けての短絡評価。つまり、拡張モジュールがfalseを返した瞬間に評価は終了。

##### state

カーネルのコネクションの追跡状態を参照。１つ以上コンマ区切りで指定する。

iptables -A INPUT **-m state --state RELATED,ESTABLISHED** # （例）

NEW 全くの新参の接続

ESTABLISHED 既に許可された接続

RELATED 新参ではあるが許可された接続

INVALID パラメータの不明な無効な接続

UNTRACKED　　 ？？？

limit

時間あたりに何パケットまで許可するかを指定

--limit rate[/second|/minute|/hour|/day]

iptables -A INPUT -s 192.168.1.0 **-m limit --limit 1/s** -j LOG --log-prefix "deny\_host: "

multiport

送信元ポートや宛先ポートの集合にマッチする。 ポートは 15 個まで指定可。 ポートの範囲指定 (port:port) は 2 ポートとカウントされる。 このモジュールが使用できるのは tcp, udp, udplite, dccp, sctp のいずれかと組み合わせた場合だけ。

iptables -A INPUT -p tcp -s $LIMITED\_LOCAL\_NET -m multiport --dports 22 -j ACCEPT

[!] --source-ports,--sports *port*[,*port*|,*port*:*port*]...

送信元ポートの指定。 ポートはカンマ区切り、コロン区切りで指定する。

したがって 53,1024:65535 はポート 53 および 1024 から 65535 までの全ポートにマッチする。

[!] --destination-ports,--dports *port*[,*port*|,*port*:*port*]...

宛先ポートの指定。

[!] --ports *port*[,*port*|,*port*:*port*]...

送信元ポートと宛先ポートの一方が指定されたポートのいずれか一つと等しければ、 マッチする。

#### パラメータ

-A,-D,-I,-R コマンドで用いられる

-4, --ipv4

iptables と iptables-restore では効果を持たない。

-6, --ipv6

　このオプションは ip6tables と ip6tables-restore では効果を持たない。

-c, --set-counters packets bytes (-I, -A, -R 操作において) パケットカウンタとバイトカウンタを初期化。

[!] -d, --destination *address*[*/mask*][,*…*]

パケットの宛先のアドレス。IPアドレスorホスト名を記述

[!] -f, --fragment IPv4のフラグ（２番目，３番目）のパケットを参照する

[!] -i, --in-interface *if\_name* パケットが入ってくるインターフェイスを指定。　if\_name；eth0，wlan0，loなど。

[!] -o, --out-interface *if\_name* パケットが出ていくインターフェイスを指定

[!] -p, --protocol *protocol*

プロトコル( tcp，udp，udplite，icmp，esp，ah，sctp，all) を指定

[!] -s, --source *address[/mask][,…*]

　パケットの送信元のアドレス。IPアドレスorホスト名を記述

　（例）iptables -A INPUT -s 192.168.1.1/24

　（例）-s 192.168.1.1/255.255.255.0 #　/24 と同じ結果になる

　（例）-s 192.168.0.0/24 # 特定のネットワークを指定

　（例）-s !192.168.0.0/24 # 特定のネットワークを指定

#### オプション

-v, --verbose

詳細な出力を行う。 list コマンドの際に、 インターフェース名、 ルールのオプション (ある場合のみ)、 TOS マスクを表示させる。 パケットとバイトカウンタも表示される。 添字 'K', 'M', 'G' は、 それぞれ 1000, 1,000,000, 1,000,000,000 倍を表す (これを変更する -x フラグも見よ)。 このオプションを append, insert, delete, replace コマンドに適用すると、 ルールについての詳細な情報を表示する。 -v は複数回指定することができ、 数が多くなるとより多くのデバッグ情報が出力される。

-w, --wait

Wait for the xtables lock. To prevent multiple instances of the program from running concurrently, an attempt will be made to obtain an exclusive lock at launch. By default, the program will exit if the lock cannot be obtained. This option will make the program wait until the exclusive lock can be obtained.

-n, --numeric

数値による出力を行う。 IP アドレスやポート番号を数値によるフォーマットで表示する。 デフォルトでは、iptables は (可能であれば) IP アドレスやポート番号をホスト名、ネットワーク名、サービス名で表示しようとする。

-x, --exact

厳密な数値で表示する。 パケットカウンタとバイトカウンタを、 K (1000 の何倍か)・M (1000K の何倍か)・G (1000M の何倍か) ではなく、 厳密な値で表示する。 このオプションは、 -L コマンドの場合のみ意味がある。

--line-numbers

ルールを一覧表示する際、 そのルールがチェインのどの位置にあるかを表す行番号を各行の始めに付加する。

--modprobe=command

チェインにルールを追加または挿入する際に、 (ターゲットやマッチングの拡張などで) 必要なモジュールをロードするために使う command を指定する。

　lo # local loop back （自分自身の事）

#### 設定の保存など

iptables の設定は電源のoff で失われてしまう。以下の方法で復元できる。

/sbin/iptables -L # ルールリストの確認

# ルールの保存はiptables-save > filename

/sbin/iptables-save > /etc/iptables/rules.v4 # コマンド実行まえにテーブル編集を済ませる事

# iptables-save > /etc/myiptables-set # こういった情報もある

# 設定の復元は iptables-restore < filename これを rc.local に書く

# /etc/rc.local （exit 0 の直前に書き込む）

# iptables-restore < /etc/iptables.ipv4.nat

#

#　自前で .sh ファイルを作成して，サービスに登録しても良いが，けっこう面倒くさい

　上記の方法は若干めんどくさい。設定保存用のパッケージも存在する。

sudo apt install iptables-persistent

iptables-persistent # 同じく，実行まえにテーブル編集を済ませる事

# 設定ファイルの保存場所

etc/iptables/rules.v4 # ipv4 版の設定

etc/iptables/rules.v6 # ipv6 版の設定

#### 具体例

基本構文

iptables 〈テーブル〉〈コマンド〉 〈マッチ〉〈ターゲット〉

ACCEPT all -- 192.168.33.1 anywhere　　　　# ⇒ 192.168.33.1 からのアクセスはすべて許可

-A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j ACCEPT

# ⇒ eth0 からのパケットをwlan0に転送し，許可されたパケットを追跡するルールを追加する。

-t nat -A POSTROUTING -s 192.168.1.0/24 -j MASQUERADE

-t nat -A POSTROUTING -o eth0 -s $internal\_ip -j MASQUERADE

# ⇒ IPマスカレード(DNAT) の設定。192.168.1.0 の外部への通信はグローバルアドレスを用いるルールを追加。

-P INPUT DROP # ⇒ 一度すべて拒否してから，

-A INPUT -m --dport 80 -j ACCEPT # 80番だけ開放する。（Htmlサーバーなどでよくある）

-A INPUT -m state NEW -m tcp -p tcp -dport 22 -j ACCEPT　# SSHを許可する

-A INPUT -m mac --mac-source 00:08:74:14:CB:10 -j ACCEPT # MACアドレス規制

資料：

<https://linuxjm.osdn.jp/html/iptables/man8/iptables.8.html> （わかりやすい。最初はこれが良い）

<https://linux.die.net/man/8/iptables> （英語）

<http://www.asahi-net.or.jp/~aa4t-nngk/ipttut/output/ipttut_all.html> （詳しいが、長い。）

参考：

<https://makezine.jp/blog/2017/09/secure-your-raspberry-pi-against-attackers.html>

iptables を用いたセキュリティ参考例

<https://qiita.com/suin/items/5c4e21fa284497782f71>

<https://kokensha.xyz/raspberry-pi/raspberry-pi-wifi-access-point/> （最強）.shスクリプト

### snort

IDS（Intrusion Detection System, 不正侵入検知システム）、IPS（Intrusion Prevention System, 不正侵入防止システム）として利用できる。（IPSはInlineモード）

### ssh

#### コマンド・オプション(ssh)

ssh-keygen # 公開鍵認証方式で使用するキーペアを生成する。

ssh-copy-id # 公開鍵をリモートホストに登録するコマンド。（使用できない環境もある）

-P “passphrase” # パスフレーズを指定 11文字以上を推奨。（10文字でクラックに約200年くらい）

# ログインパスワードではない。zipファイルにかかっているパスワードの様な物。

-i　“SomeDir\FileName” # 秘密鍵ファイルの指定

-F # 設定ファイルの指定。デフォルトでは $HOME/.ssh/config が使用される。

-l # ログインユーザー名を指定　？？？使う事あるの？

#### 暗号鍵の概要

公開鍵：サーバー側に持たせておく鍵。　.pub の方。

作ったときには「id\_rsa.pub」という名前で作成され、基本的には「authorized\_keys」という名前にリネームする。

複数の公開鍵を作る場合は、authorized\_keysに追記していく

※Windowsの場合は %USERPROFILE%\.ssh\authorized\_keys

Linuxは $HOME/.ssh/authorized\_keys に配置する。

秘密鍵

クライアント側（個人や特定の機器のみ）で保有しておく鍵。**だれにも渡してはいけない。**

ファイルの中身だけが重要で、ファイル名は変更しても良い。

・逆計算が困難な数学上の問題を利用することで、ペアとなる鍵の片方（公開鍵）を公開したとしても、秘密鍵が事実上割り出せなくなる。

・公開鍵設定しても、パスワード認証は有効。サーバー側のsshd\_configを変更してパスワード認証を無効にする。

（参考）<https://qiita.com/angel_p_57/items/897bf94160be8d637585>

#### 鍵の作成手順

・１　サーバーにログイン。リモート（SSH）でも直接でも良い。

（補足）クライアント側で作って、公開鍵をサーバーに転送しても良いらしい

・２ サーバー側で鍵を作る

ssh-keygen -t ed25519 # 公開鍵暗号を作成する。　デフォルトでは /.ssh/id\_ed25519 となる

# id\_ed25519 と id\_ed25519.pub の２つが作成される。

ssh-keygen -t rsa -b 4096 # オプション -b でビット数の指定ができる。

・パスフレーズは必須ではない。-P “” など空白の指定でスキップできる。指定した場合は忘れないようにする。

・鍵が作成される場所　　windows：c:\users\[ユーザー名]\.ssh Linux：$HOME\.ssh

・３ 公開鍵の設定

２で作成された公開鍵（.pubの方）の名前をauthorized\_keysに変更。パーミションを600(所有者rw-) に。

２で作成されたフォルダ $HOME\.ssh のパーミッションを700(所有者rwx) に。

cd ~/.ssh/ #

mv id\_rsa.pub >> authorized\_keys # ファイル名変更（ファイル名は何で暗号するかで異なる）

chmod 600 authorized\_keys #

chmod 700 ~/.ssh #

・４ 秘密鍵の転送

scp(WinSCPなど)　を用いてファイルを転送する。

scp [ユーザー名]@[ホスト名]：ファイルパス

Windows　の場合（Comand Prompt）、scp コピー元 コピー先 としないと動かない。

・５ クライアント側から接続

ssh の　-i オプションで暗号鍵を指定する。

秘密鍵のファイル名、パスは自由に変更可能

ssh -i ~/.ssh/[鍵ファイルの名前] hoge@example.com # Linux系の場合

ssh -I .ssh/[鍵ファイルの名前] hoge@example.com # Windowsの場合（usersフォルダはpath通っている）

#### 設定ファイル(sshd)

/etc/ssh/sshd\_config　# ssh daemon の意らしい。

※Windowsの場合は%PROGRAMDATA%\ssh\sshd\_config

Port 22 # SSHのポート番号を変更。49152～65535あたりで好きなポート番号を選ぶ。

PermitRootLogin no # root ユーザーでのssh接続を有効／無効に。

PubkeyAuthentication yes # 鍵認証を有効化

PasswordAuthentication no # 平文パスワードでのログイン禁止（真ん中より少し上）

PermitEmptyPasswords no # 一応、空パスワードでのログインも明示的に禁止

sudo systemctl daemon-reload　 # 編集終わったら、デーモンの再起動

sudo systemctl enable ssh # SSHサーバーの自動起動設定

sudo systemctl start ssh # SSHサーバーの再起動

sudo /etc/init.d/ssh restart # もしくはこれ。ただ、init.d はその内廃止されるらしい 19.10.31

#### 設定ファイル(ssh)

/etc/ssh/ssh\_config # 外のサーバーに接続する際の設定ファイル。

ログイン情報を残す事になるので、自身で十分考える。

別資料：ソフトウェア情報 - OpenSSHに移行

# Command Prompt（Windowsからの接続）

**::** ssh ログインユーザー名@ホストIP/SSID　　でログインできる。

ssh -p 1234 -i key pi@192.168.1.15 **::** 接続

exit **::** ssh接続の終了

#### known\_hosts

一度接続したことのあるサーバーのSSHサーバ証明書は以下の場所に格納されている。

linux： $HOME/.ssh/known\_hosts

windows： users\[ユーザー名]\.ssh\known\_hosts

（参考）<https://qiita.com/wnoguchi/items/a72a042bb8159c35d056>

（公式ドキュメント？）<https://euske.github.io/openssh-jman/ssh_config.html>

（これも公式？）<https://www.openssh.com/manual.html>

（誤解？）<https://qiita.com/angel_p_57/items/2e3f3f8661de32a0d432>

### VNC

Raspberry Pi OSのVNCサーバーはRealVNC。その為，クライアントアプリはVNC Viewerが良い。

一方，RDP（Remote Desktop Protocol）にも対応している。

sudo apt-get install xrdp

sudo service xrdp restart

Win10であればRDPは最初から入っている。

Win10用のVNCクライアントではUltraVNCがよく紹介されているが、プロトコルの違いでラズパイには接続できない。

### PPPoE

インターネットプロバイダに直接接続する際、など。

sudo apt install pppoe pppoeconf # パッケージの追加

# ここでプロバイダのモデムと接続

sudo pppoeconf

ip addr show ppp0 # インターフェイス情報を見る　IPアドレスが割り振られていたら成功

pon dsl-provider # 接続を開始

poff # 切断

# /etc/ppp/options # pppoe 設定ファイルの場所

# /etc/ppp/pap-secrets # プロバイダとの接続情報など

# 注意点　dhcpcd.conf でeth0に固定 IP を割り振るとppp0の方で接続できなくなる。

### ローカルDNSサーバー

# /etc/dnsmasq.conf

domain-needed # 「.」またはドメインを含まない名前は上位 DNS にフォワード(転送) しない

bogus-priv # プライベート IP は上位 DNS にフォワード(転送) しない

local=/my-local-domain.com/ # ローカルドメイン設定

expand-hosts # ドメイン自動補完 :「domain=」にドメインが設定された場合に有効

domain=my-local-domain.com #

（参考）

<https://server.etutsplus.com/dnsmasq-setup-internal-dns/>

### DDNS

Dynamic Domain Name System

ddclient ：DDNSクライアント。対応していて使えそうなDDNSサービスはno-ipくらい？

### 参考情報

<https://qiita.com/JhonnyBravo/items/d8445d21d9c79501c6f6>

#### SSLサーバー証明書

#### Brython

Python → JavaScriptのトランスパイル。brython.js；brython\_stdlib.js

結構めんどくさいらしい。

## データベース

### postgresql

※別ドキュメント「ソフトウェア情報」へ移動中

#### 概要

オープンソースのデータベースで最も有名で，有料製品とくらべても遜色が無いと言われている。

ここではラズパイでpsqlを起動し，クライアント機のGUIから接続するまでの手順を記す。

注意点として，クライアント側にもドライバのインストールが必要

Windows PostgreSQL ODBC Driver <https://www.postgresql.org/ftp/odbc/versions/msi/>

Linux postgresql-client-[ver] aptやrpm でインストール可能

#### ロールの管理

コマンドは大文字でも小文字でも良い。また，最後に「；」が必要な点に注意。

（補足）新規にコマンドを打つ前に「；」を空打ちした方が良い。（それまでの打ち間違いをリセットする目的）

=> CREATE ROLE *user1* WITH login password '*password*'**;** # 最後の「；」に注意！

# WITH以下はオプションを記述。WITHはあってもなくても可。

# オプションは大文字でも小文字でも良い。

=> ALTER ROLE user1 createdb; # 後でロールの属性を変更する場合。

=> \du # ユーザー一覧（ 元の画面に戻るには q を押す）

参考）<https://www.postgresql.jp/document/9.6/html/sql-createrole.html>

#### データベースの管理

ユーザー名と同名のデータベースが無いとログインできないのせ作成して置く。

=> create database *user1* owner *user1*； # データベース作成

=> \l # データベースの一覧

=> drop database *dbname*; # 削除

# postgres，template0，template1 これらはPostgreSQLに必要なデータベース。削除できない。

⇒ この辺の仕様がどうなっているかは再調査が必要 2021-03-10

#### 設定

ロケールの修正？(2021-03-10)

sudo service postgresql stop # サービスを停止

sudo rm -rf /var/lib/postgresql/[ver]/main # ファイルを削除

su – postgres # ユーザーを切り換え

PGDATA=/var/lib/postgresql/[ver]/main /usr/lib/postgresql/[ver]/bin/initdb --no-locale -E UTF-8 # UTF8, ロケールなしで初期化

sudo service postgresql start # サービスを再度起動

（情報元）<https://qiita.com/mnbszk/items/7d8f01afb827f0872328>

その他

sudo service postgres stop # 停止 開始はstart

psql -l # データベースの確認（psql 起動していない状態で）

#### 外部からの接続

1. ・まずはラズパイ側で設定ファイルを変更。 （参考）[設定](#_設定)
2. ・ラズパイ側：　psql -V でバージョン確認。（--version でも良い）

一般ツールのインストール方法

Ubuntu

sudo apt install postgresql-client-[ver]

# psql -h ホスト名 -p ポート番号 -U ロール名 -d データベース名

psql -h 192.168.1.30 -p 5432 -U pi -d pi

Windows

（ダウンロード先）<https://www.postgresql.org/ftp/odbc/versions/msi/>

・管理ツール　スタート → Windows管理ツール → ODBCデータソース(32bit)

ユーザーDSNタブ： → PostgreSQL UTF8※

設定 （補足）サーバー名：IPアドレス等を入れても良い

接続確認 構成 → 

※ANSIかUTF8かはODBC ドライバを使用する Windows アプリケーションのビルドオプションと合わせる。

#### GUI(DBeaver)

（ダウンロード）<https://dbeaver.io/>

Windows zip版とインストーラ版がある。

Ubuntu(Debian) .debファイルをダウンロード → dpkg -i でインストール。

※他にもHeidiSQL，pgAdminなどが人気。

#### pythonから使う

sudo pip install psycopg2 # pipでインストール

# sudo apt install python3-psycopg2 # apt で入れる情報があるが？？

sudo pip install psycopg2

## .NETアプリを使う

mono をラズパイにインストール

sudo apt install mono-complete

Visual Studioでは普通に.NETアプリを作成（.NET Coreでは無い事に注意！）

Formアプリも作れる。恐らくこれがラズパイでGUIアプリを作る一番容易な方法。

## その他

・sudo apt install postgresql-client # クライアント機能のみのpostgre

apacheでもDNSを作れる？

vsftpd ：ftp通信アプリ。winで言うffftp

pi-bluetooth

ブルートゥース管理パッケージ？

# 資料

## 公式オンラインドキュメント

情報はできるだけ原典に近いものから取得する

RaspberryPi公式資料：<https://www.raspberrypi.com/documentation/>

OS公式資料：<https://www.raspberrypi.com/documentation/computers/os.html>

Linux 系コマンド：

Debian公式：<https://www.debian.org/>

Python公式：<https://docs.python.org/3/> 　（日本語も選択可。ただし訳が変）

PySerial公式：[https://pythonhosted.org//pyserial/](https://pythonhosted.org/pyserial/)

Apache公式サイト：<https://httpd.apache.org/docs/2.4/ja/>

Django公式チュートリアル：<https://docs.djangoproject.com/ja/2.2/>

PostgreSQL（英語オリジナル）：<https://www.postgresql.org/docs/>

PostgreSQL（日本の協会）：https://www.postgresql.jp/document/

OpenSSH（英語オリジナル）：<http://www.openssh.com/>

OpenSSH（日本語）：<https://euske.github.io/openssh-jman/ssh_config.html>

Github <https://help.github.com/ja>

## Ping-T

Ping-TはCCNAやLinuC などの資格の独学サイト。Linuxの仕組みやコマンドに関する情報が分かりやすく記載されている。Raspberry Pi OSの事を調べるより，Linux自体の操作方法を覚えた方が効率的な面もあり，Linuxの資格には興味なくても，試してみる価値はある。

<https://ping-t.com/>

最初にやるなら：Linux Essential

次にやるなら：LinuC/LPICコンテンツ の 最強WEB問題集LPIC Lv1-101