最低限これだけ設定すれば、とりあえず動かすことが出来る！

はじめての RaSCSI 導入マニュアル

2018/12/25 版　Project M.P.S

～序章～

RaSCSI とは、GIMONS 氏(Twitter：@kugimoto0715)が開発した、Raspberry PI を使い SCSI デバイスをソフトウェアでエミュレートし、実機からは実際に SCSI デバイスが繋がっているように振舞う、長年ストレージの問題に悩まされてきた、レトロPCにとっては革新的なプログラムです

「原型」は 2017 年始め頃に動き出し、多くの有志による試行錯誤を経てソフト・ハード共にバージョンアップを重ね、約半年の時を経て現在の潮流が形作られました

最初のバージョンリリースから 2 年の節目を迎え、Raspberry PI の信号を SCSI 信号に変換するハードウェアは一種の到達点を迎えましたが、実際に RaSCSI を使えるようになるまでのソフトウェア面のサポートは、いくつかの導入解説サイトは存在しますが、これらはある程度の Linux の知識を持った上で成り立つものであり、全く Linux を触ったことが無いユーザーにとっては非常に敷居が高いものです

このマニュアルは、全くLinuxを触ったことが無くても、**「この通り設定すればとりあえず RaSCSI を動かすことが出来る」**を目指しております。ただし、実際に Raspberry PI を操作するターミナルプログラムなどを最低限操作する知識は必要になりますのでご了承ください

～1 章　RaSCSI 変換アダプタから見るバイナリプログラム選び～

RaSCSI 変換アダプタは初期の試行錯誤からなる、各々の GPIO ピンや信号の最適化の関係で複数のバイナリプログラムが存在し、ちょっとした混乱の元となっています。ある程度他のRaSCSIアダプタを見てきた熟練者ならば、アダプタを見ただけで「これは〇〇版のバイナリプログラムを使う」と分かるのですが、初めて RaSCSI というものに触れた初心者にとっては、いま手元にある変換アダプタが何版のバイナリを使用するかを窺い知るのは困難です

RaSCSI プログラムには以下のバイナリプログラムが収められています
・GAMERnium 版
・あいぽむ版
・スタンダード版
・フルスペック版

これらはプログラムの中身は違えどファイル名は同じなので、アダプタの種類とバイナリプログラムがミスマッチしていると、RaSCSI プログラムとして表面上は動いていても、SCSI デバイスとしてはコンピューター側で認識されず、何が原因か分からないまま頭を抱えることになります(後述するカーネルモジュールのミスマッチでも同様です)

現在、一般に頒布された RaSCSI アダプタは以下のようなものがあります

GAMERnium.com 版　http://gamernium.com/archives/tag/rascsi

Version 1

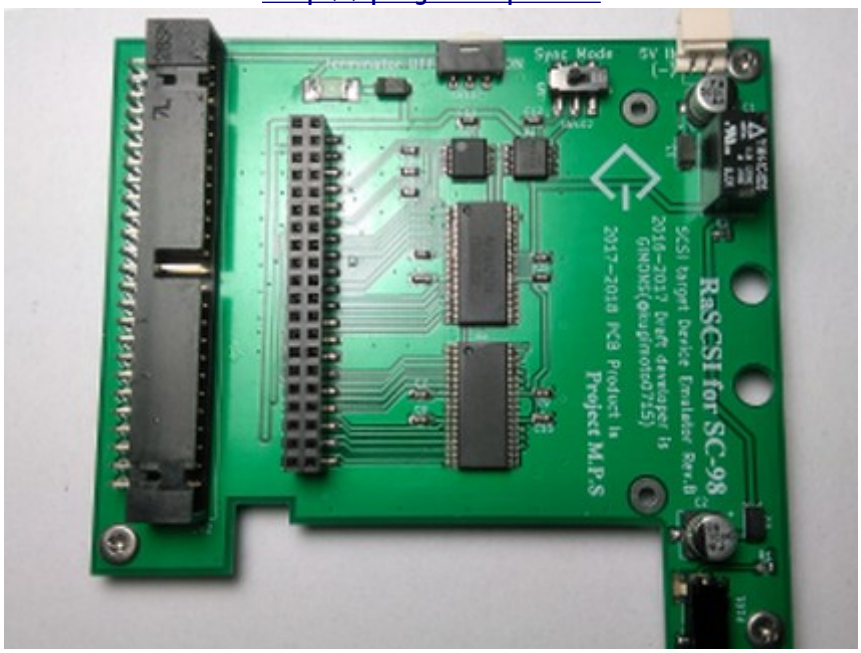
Version 2

「GAMERnium 版」バイナリを使用

あいぼむ版「RaSCSI Adapter」
http://products.aibom.net/rascsi/

「あいぼむ版」バイナリを使用

Project M.P.S 版「RaSCSI for SC-98」
http://projectmps.net



「あいぼむ版」バイナリを使用

GAMERnium.com 版とあいぼむ版が実店舗展開での比較的早い時期にリリースされたため、この 2 つが事実上の RaSCSI アダプタ・スタンダードとなっています

双方ともスタンダード版からカスタマイズされた GPIO ピンと信号を使用しているため、別々のバイナリを使用します

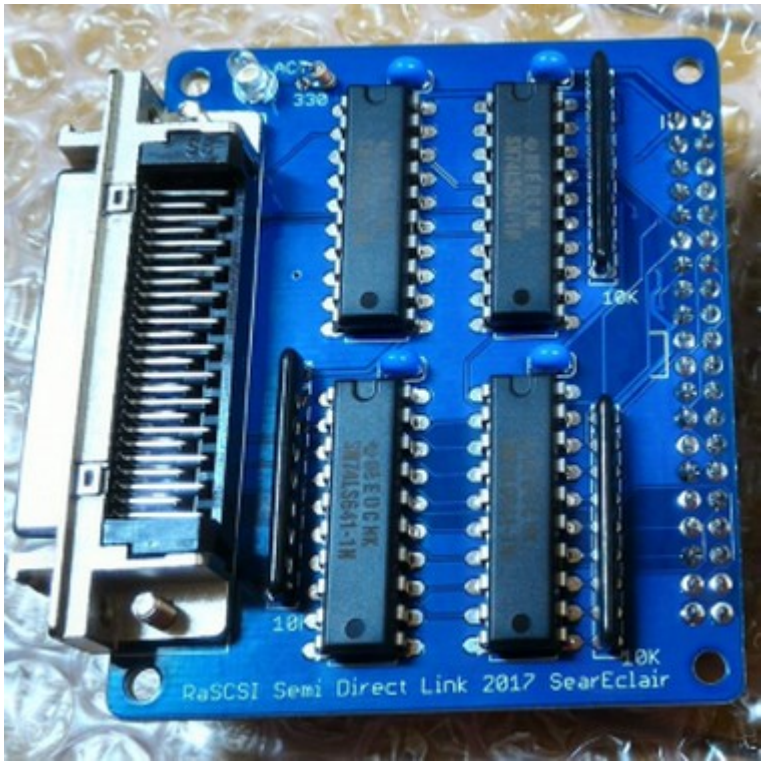
RaSCSI for SC-98 は後発ですが、あいぼむ版と同じ制御チップを使っているため、あいぼむ版互換になっています

えくしみえむ版

http://www.katch.ne.jp/~x680x0_mania/rascsi_pcb.html

焦がしエクレア版

Twitter：@SearEclair

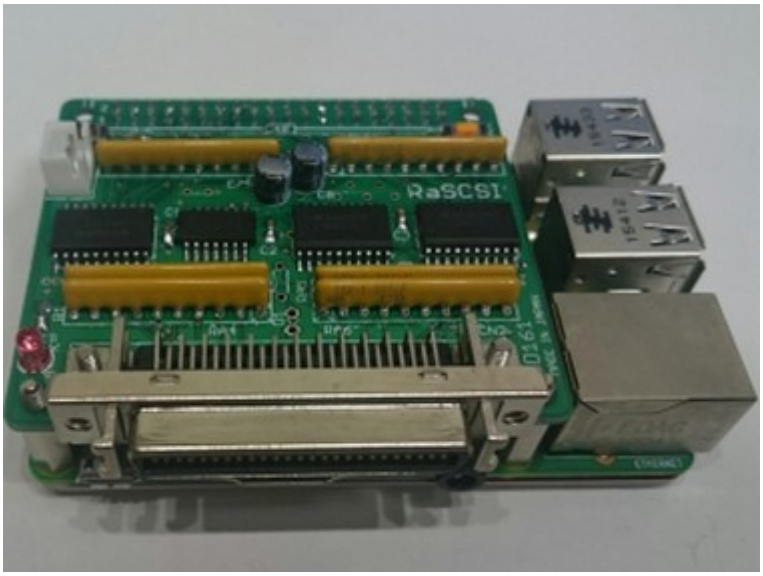


あいぼむ版
「RaSCSI Adapter Lite」

http://products.aibom.net/ral/

mobile 版
「ND161」

https://www.ayuca.jp/rascsi/



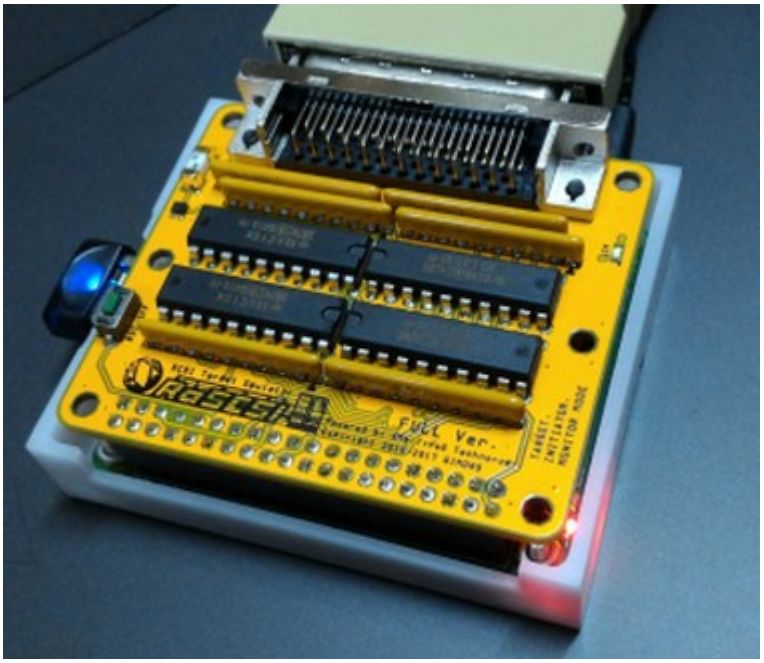
これらは全て「スタンダード版」のバイナリを使用します

この４種類の中で「焦がしエクレア版」と「mobile 版」は基本的に実店舗展開をしていない、希望者のみ頒布バージョンの為、出回り数は多くなく、あまり見かけることは無いと思われます

ちょっと大変なのが「えくしみえむ版」で、製作者ロゴ入り基板が頒布されたほか、設計データが公開されているため、基板のレイアウトを踏襲した基板の色違いバージョンが流通している可能性があります。見分け方は頒布公式ページに各種基板レイアウトがありますので、それで手元の基板と確認をしてください

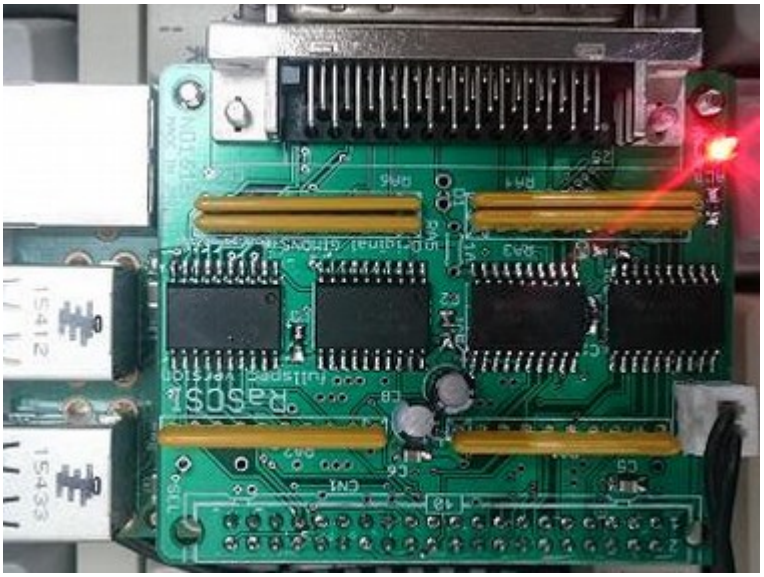
えくしみえむ版

http://www.katch.ne.jp/~x680x0_mania/rascsi_pcb.html



mobile 版「ND161B」

https://www.ayuca.jp/rascsi/

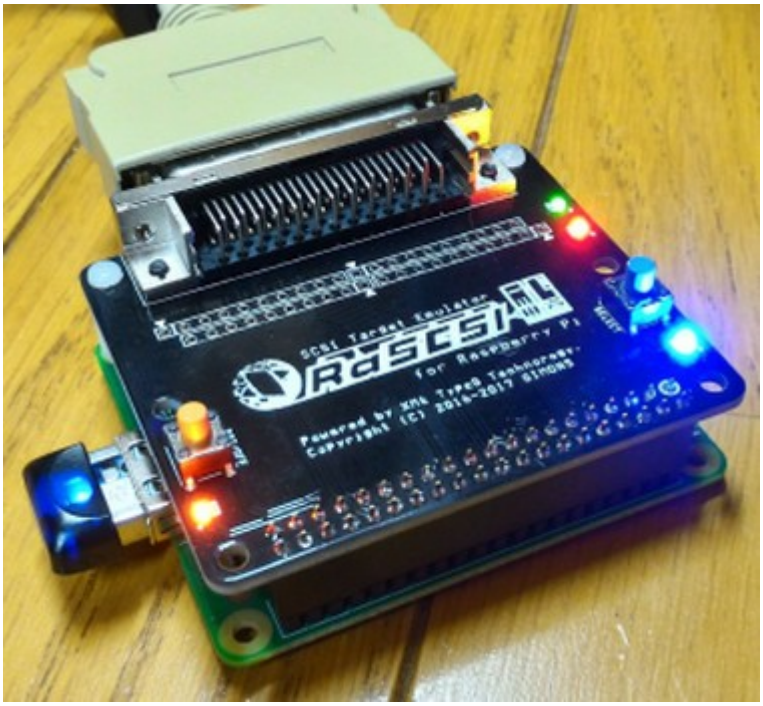


Project M.P.S 版「RaSCSI ZERO」 http://projectmps.net

これらは全て「フルスペック版」のバイナリを使用します

えくしみえむ版はスタンダード版と見分けがつきにくいですが、スイッチの数が減っていたり、LED 周りが変わっていたり「FULL Ver」のシルクが追加されていたりしますので何となく分かると思います。こちらも設計データが公開されているため、色違いの派生が存在する可能性があります

mobile 版は 2 番目のチップが変わっていることと「ND161B」という型番及び「fullspec version」というシルクがありますので、そこから見分けが付くと思います



その他「**直結系**」、上 2 つがえくしみえむ版、下が焦がしエクレア版です。このほかにも存在します

RaSCSI の黎明期に試行錯誤の一環として製作されましたが、Raspberry PI の電圧(3.3V)とSCSIの電圧(5V)を同時に混ぜると電気的によくないという観点から、後に電圧レベル変換チップを使用するバッファー式に移り変わっていきました

実際、これらの直結版は変換をかまさないのでパフォーマンスが出るとか、変換チップを使わないのでコスト的に有利だとかいわれていますし、直結したRaspberry PI やSCSIボードが壊れたという話も今のところ聞いた事が無いので、もしかしたらこの方式も「アリ」なのではないかと思っています

これらの基板を今から使う機会があるとは思いませんが

バイナリは「フルスペック版」を使用します

以上、今手元にある RaSCSI アダプタが何版のバイナリを使うか分かりましたか？これらの情報は後々必要になってきますので、覚えておいてください

次は Raspberry PI 用の OS である「Raspbian」のダウンロード～インストールから、最初に行うべき設定などについて説明していきます

～2章　Raspberry PI 用 OS「Raspbian」のインストールと初期の設定～

それこそコンピューターというものが個人の手で取り扱えるようになった昔から「コンピューターはソフトが無ければただの箱」という名言(?)がありまして、それは Raspberry PI に対しても多分に洩れず、OSをインストールしなければただの無駄に電気を食うコンピューターに過ぎません

ただし、コンピューターに標準で OS などがインストールされていなかった当時と今の大きな違いは「OS が無料で手に入る」(もちろん通信費とか電気代は別ですよ？)ことと、インターネットの発達で「目的に対する情報が多すぎる」ことです

昔は少ない情報をかき集めて目的を達成していましたが、今は逆に多すぎる情報を取捨選択して目的を達成する時代です。情報リテラシーが高い人ならば、膨大な情報の中から必要な情報を探し出して応用することは容易ですが、今まで使ったことの無い電子機器を目の前に、必要な情報を探そうとして情報の波に飲まれ**「結局最初に何をしたらいいのか分からない」**と呆然と立ち尽くすことも少なくありません

そんな状況を少しでも手助けすべく、手順をなるべく分かりやすく解説していきたいと思います

最初に必要な物

・Raspberry PI(出来れば 3B 以降)
※基本的に通常版と小型のZeroは共用できますが、一部 Zero 専用の RaSCSI アダプタもあります

・microSD カード 8GB～
※通常タイプの Raspberry PI をシステム用途で使う場合は 8GB、Zero など外部 I/O が少ない場合は 16GB を推奨します。まあ、大は小を兼ねるという事で、8GB 以上あれば好きな容量の microSD カードを使ってください

・microSD カードが読めるリーダー

・給電用 microUSB アダプタ 2A 以上
※電力不足が思わぬトラブルの原因になることがあります、そうならないためにもガッツリ大容量のACアダプタを

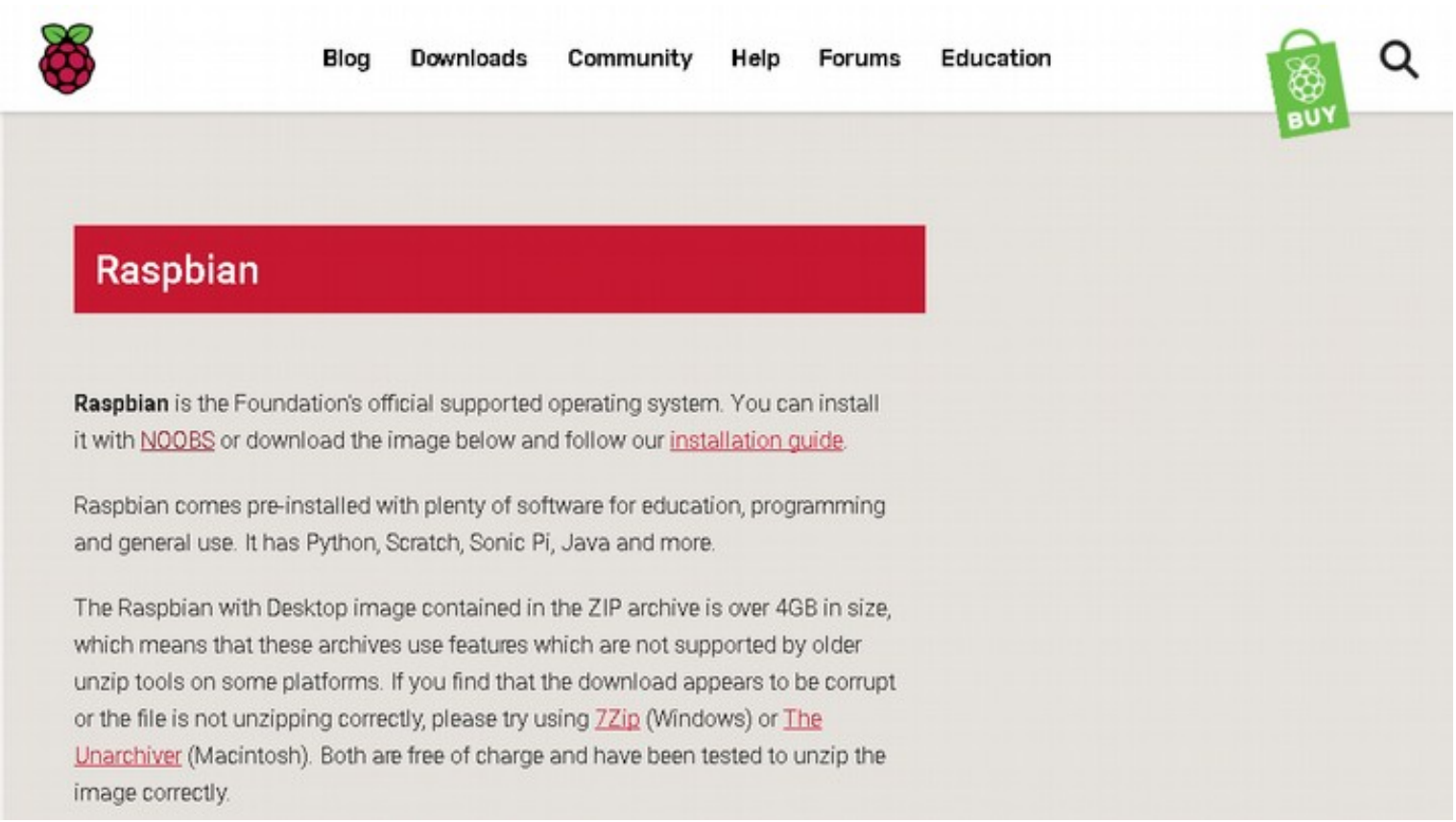
・ルータを介したネットワーク環境
※コレが無いと wi-fi や有線LAN接続が出来なくなり、初っ端から詰みます

・RaSCSI アダプタや SCSI ケーブルなど一式
※まぁ、そもそもこの説明書自体がRaSCSIの導入書なので・・・

なお、本来なら Raspberry PI 上で直接操作するために HDMI 接続のケーブルやらモニタ、そして USB キーボードなどが必要になりますが、今回はネットワークを介して全ての設定を行うため必要ありません

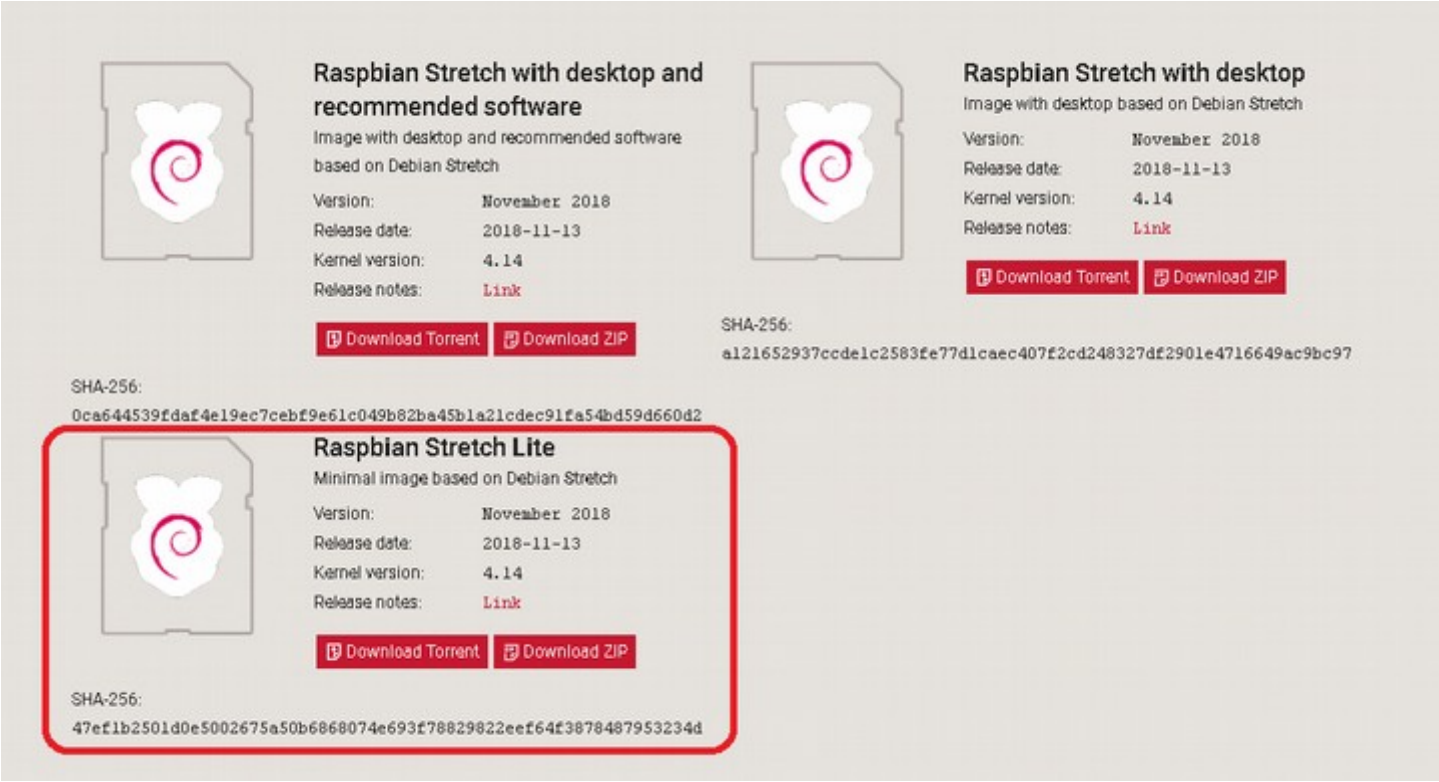
Raspbian のダウンロード

「raspbian」で検索するか、https://www.raspberrypi.org/downloads/raspbian/ をクリック or コピー＆ペーストして、raspbian の配布サイトまで飛んでください



いきなり英語まみれで、英語アレルギーの人などはドン引き画面ですが、大したことは書いていないのでスルーしてください

少し下にスクロールすると、3 つのダウンロード選択肢が現れます



左上から右に
左：GUI 画面に最初からたくさんアプリが付いているよ！オススメ！
右：GUI 画面の華やかな画面で初心者にも操作が優しいよ、アプリは後で各自でインストールしてね
下：Raspberry PI を動かすためだけの最低限の機能しか入っていないよ

とまあ、適当に意訳するとこんな感じです。GUI 画面や華やかなアプリは初心者には大変有難いのですが、引き換えに大量のディスクスペースと少なくないCPUパワーとメモリを持っていかれるので、基本的にバックグラウンドモードで動作し、GUI 画面も見ないし、アプリも使うことが無い RaSCSI プログラムにとっては全く不要です

ということで、3 つ目の「Raspbian Stretch Lite」をダウンロードします。「Download Torrent」は P2P ソフト「BitTorrent」用のファイルで、P2P 技術を使うためサーバの負荷が低くて転送速度が速いのですが、当然ながら PC に BitTorrent をインストールしたり、必要に応じてポートを空けたりしないといけなくて面倒なので、サーバからの一括ダウンロードである**「Download ZIP」**を選択してダウンロードを行ってください

容量は 300MB 台、転送速度はあまり早くなくて時間によってはダウンロード完了までに 20 分以上かかるので、一息ついて飲み物でも飲んで気長に待ちましょう

ダウンロードが終わったらダウンロードフォルダに**「yyyy-mm-dd-raspbian-stretch-lite.zip」**というファイルがありますので、それをどこか空きスペースがある場所に展開してください(展開すると 1.7GB ほどの容量があります)

※ファイル名の yyyy-mm-dd はリリース日付名です、上の画像に倣うなら 2018-11-13-raspbian-stretch-lite.zip

microSD カードの用意とダウンロードしたイメージデータの書き込み

「ブートできる(micro)SD カードを作れるイメージデータ書き込みソフト」は、これといった定番は無いのですが、Raspberry PI のイメージデータ書き込みで検索してみると**「Win32DiskImager」**が割と定番のようです(Windows 用)

Win32DiskImager

https://sourceforge.net/projects/win32diskimager/

例によって英語ページです。このテのページは本筋とは関係ないところにダウンロードボタンがあったりして、よく分からないままボタンを押すと目的とは違うプログラムがダウンロードされたり、実はリンクで変なところに飛ばされたり、場合によっては悪意のプログラムによりウイルスに感染するタチの悪いものまであります

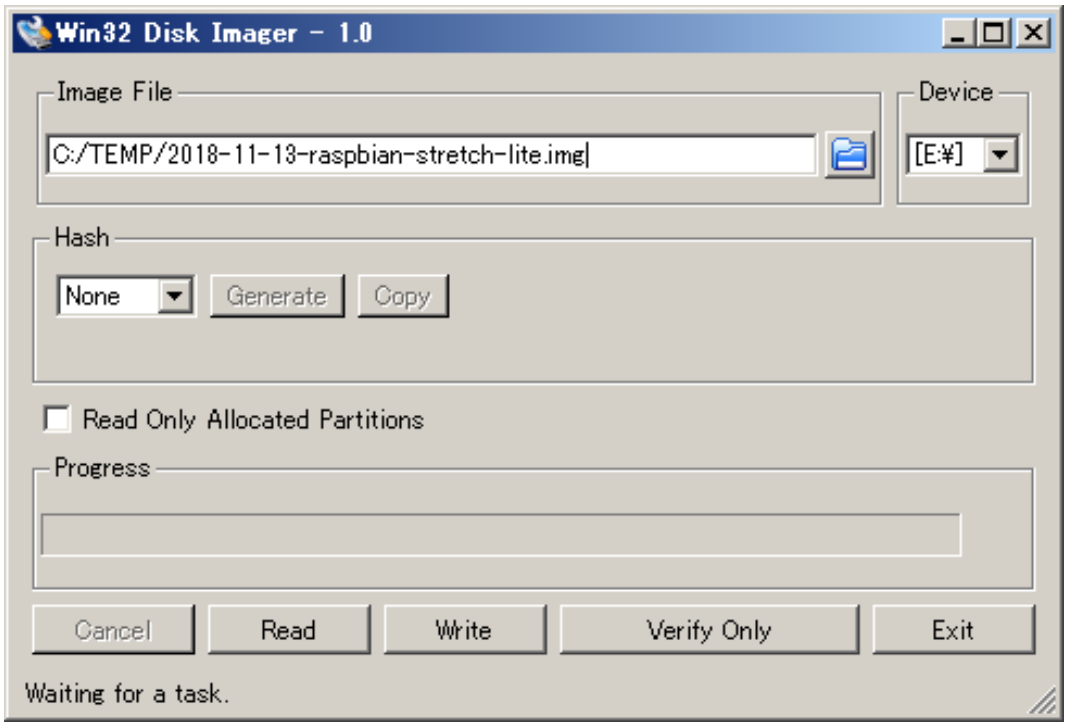
SourceForge.net はオープンソースプロジェクトの総本山みたいな所なので、そのテには厳しいですが、こういう造りの Web ページには一定の警戒感が必要です

ダウンロードですが、でかでかと緑色で「Download」と書かれているボタンがページの一番上にありますので、これをクリックしてください。ブラウザ側のダウンロードウィンドウが開きダウンロードが始まります

ただし、EXE 形式のファイルがそのままアップロードされているので、ウイルス検知ソフトによってはダウンロードをブロックされるかもしれませんので注意してください

win32diskimager-1.0.0-install.exe というファイル名でダウンロードされます。インストーラー形式なのでそのまま実行してインストールを進めてください

コンピューターに microSD カードを接続します。接続を確認したら、Win32DiskImager の実行ファイル「**Win32DiskImager.exe**」を実行します。システムのセキュリティ状態によっては実行時に確認のメッセージが出る場合があります

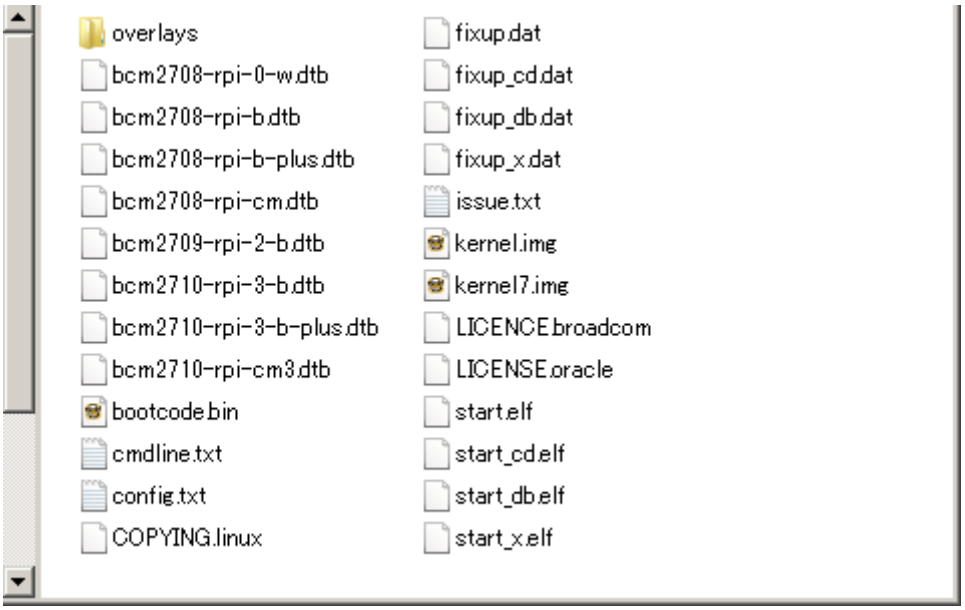


「**Image File**」が先ほどダウンロードした Raspbian の圧縮ファイルを展開したイメージファイルデータ

「**Device**」が microSD カードが挿さっているドライブです。このドライブは microSD カード限定ではなくリムーバブルデバイスである USB メモリなども対象になっていますので、複数のデバイスが選択できる場合はイメージを書き込みしたい microSD カードが挿さっているドライブを指定してください

イメージやドライブなどの確認が終わったら**Writeボタン**を押してください。書き込みが始まります

書き込み終了のダイアログが出たらプログラムを終了してください。microSD カードはまだ抜かないでください



書き込みが終わった microSD カードが挿さっているドライブを開いてみてください。Raspbian のバージョンによって違いはあるかもしれませんが、大体こんな感じのファイル構成になっていると思います

これで microSD カードへの Raspbian イメージデータの書き込みは終了です。このままこの microSD カードを Raspberry PI に挿せばシステムを起動することが出来ますが、設定を変更するのに HDMI 接続のモニタや USB 接続のキーボードやマウスが必要になってくるので、もうひと手間加えます

モニタや入力デバイスを使わないでネットワークや SSH を設定する

本来ならここで Raspberry PI に HDMI ケーブルや USB ケーブルを繋いでモニタを見ながら設定をするのですが、そのためだけにケーブルや入力デバイス用意するのは面倒ですよね？(Raspberry PI Zero では変換コネクタもいりますし)

そこで最初から SSH 接続による Raspberry PI の設定が出来るように設定します

SSH 接続の有効化

Windows 標準の「メモ帳」を開き、何も書かないまま「ssh」というファイル名で microSD カードが接続されているドライブに保存する。その際ファイル名が「ssh.txt」となっているので、txt の部分を消して ssh(拡張子無し)の状態にする

※Windows のエクスプローラ設定で必ず「拡張子を表示する」設定を行ってください

ネットワーク接続の設定

これがクセモノです、なぜならネットワーク接続の方法が複数あるからです

「有線LAN接続」と「無線LAN接続」
「DHCP による IP アドレス取得」と「固定 IP 割り当て」
「無線の 2.4GHz 接続」と「無線の 5GHz 接続」

まず簡単なほうから。無線 LAN 環境で主に 5GHz で運用している場合は、Raspberry PI に有線LAN接続するか無線ルータ側に 2.4GHz の電波を出すように設定してください。色々な事情でどうしても 2.4GHz の設定をしたくないし、有線での接続もしたくない場合は、最新版の「Raspberry PI 3B+」を使用してください
(もしくは USB に 5GHz 対応の無線LANアダプタを接続する)

1：有線LAN＋DHCP 接続の場合→Raspberry PI に LAN ケーブルを繋ぐだけで OK です

2：無線 LAN＋DHCP 接続の場合 ↓こちらを参考にしてください
https://qiita.com/mascii/items/0d1a280ac58ed8f6f999

具体的には、メモ帳を開いて

```
country=JP
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
    ssid="test"
    psk="abcd1234"
}
```

上記の文字列をそのままメモ帳にコピー&ペーストし、ssid と psk(暗号化パスフレーズ)を各自の無線LANルータの設定に合わせて**「wpa_supplicant.conf 」**というファイル名で、microSD カードが接続されているドライブに保存する
(そして txt 拡張子を消す)

パスワードを平文で保存したくない場合は、リンクの「256bit のハッシュ化されたキーを保存する場合」を参照してください

3：有線・無線LANで固定 IP 接続の場合

残念ながら、事前設定する方法が分かっておりません。一度DHCPによるIP割り当てで SSH 接続するか、RaspberryPI に直接モニタと入力デバイスを繋いで、以下のサイトを参考にして固定 IP を割り当ててみてください
https://qiita.com/momotaro98/items/fa94c0ed6e9e727fe15e

これで Raspberry PI にモニタや入力デバイスを繋がずに、ネットワーク経由でSSH接続をする準備が整いました
SD カードリーダーから microSD カードを取り外し、Raspberry PI のカードスロットに挿入してください

Raspberry PI に SSH 接続をして色々な設定をする

ようやく舞台が Raspbeery PI 上に移ります。RaSCSI プログラムが使えるようになるのはもう少し先です

SSH クライアントは何にしよう？

Windows で使える SSH クライアントは以下のサイトが参考になります
https://weblabo.oscasierra.net/ssh-software/

鉄板は「TeraTerm」、あとは「PuTTY」が海外産ながら、日本語化パッチも出ていて人気が高いようです

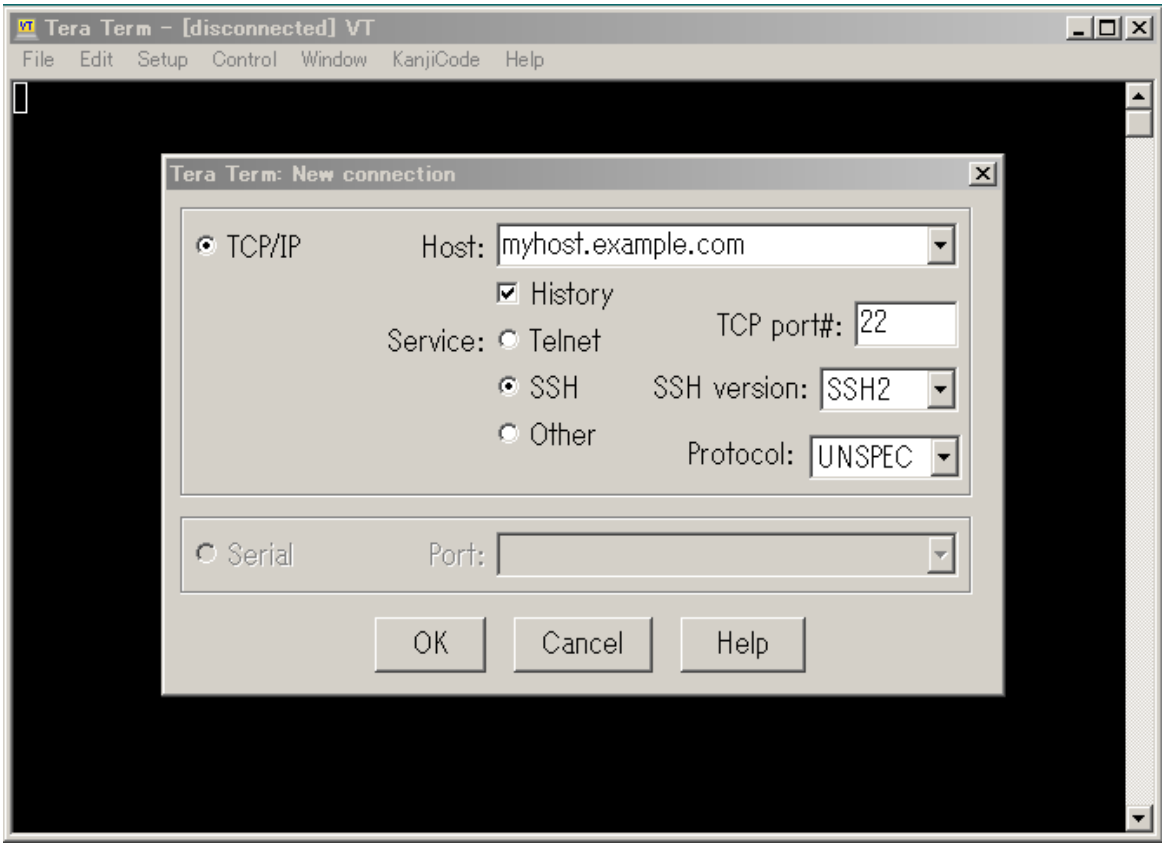
Tera Term
https://ja.osdn.net/projects/ttssh2/

PuTTY
https://www.putty.org/
非公式日本語化版
https://hp.vector.co.jp/authors/VA024651/PuTTYkj.html

ここでは Tera Term を使って説明を進めていきます

Raspberry PI への SSH 接続と最初にやること

まず、RaSCSI アダプタを接続しない状態で、Raspberry PI に microUSB 電源給電コネクタを挿し起動させます。アクセス LED の点滅が落ち着いたら、TeraTerm(ttermpro.exe)を起動させます



続いて Raspberry PI に割り当てられている IP アドレスを探します。本来なら Raspberry PI をモニタに繋いで、接続されたキーボードから「ifconfig」コマンドで自 IP アドレスを知るのですが、最初からモニタレスで SSH 接続を前提にする場合はその方法が使えませんので、**Windows のコマンドプロンプトから ping コマンドで総当りをします**

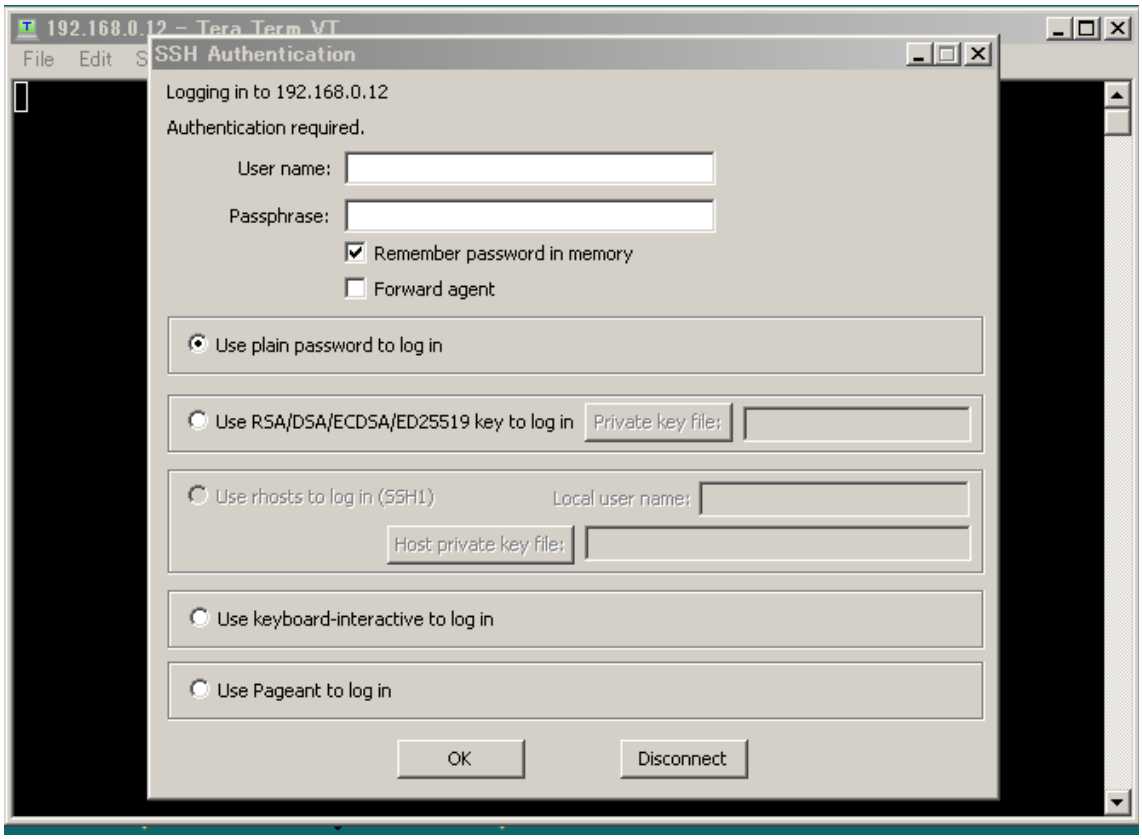
まず、Windows のコマンドプロンプトを開き「**ipconfig コマンド**」にて、現在のマシンの IP アドレスを確認します
コマンドを入力すると、色々な数字がずらずらっと列挙されますので、その中から「**IPv4 アドレス**」という文字列を探し出し、その先にある数字が現在のマシンの IP アドレスです(特殊な設定をしていなければ 192 で始まることがほとんどです)

基本的に後から起動したネットワークデバイスが、すでに起動しているネットワークデバイスの IP アドレスの前に来ることはまずないので、そのマシンの IP アドレスを起点として、**ping コマンド**で末端の数字を 1 づつ増やしてエコーが返ってくるアドレスを探します

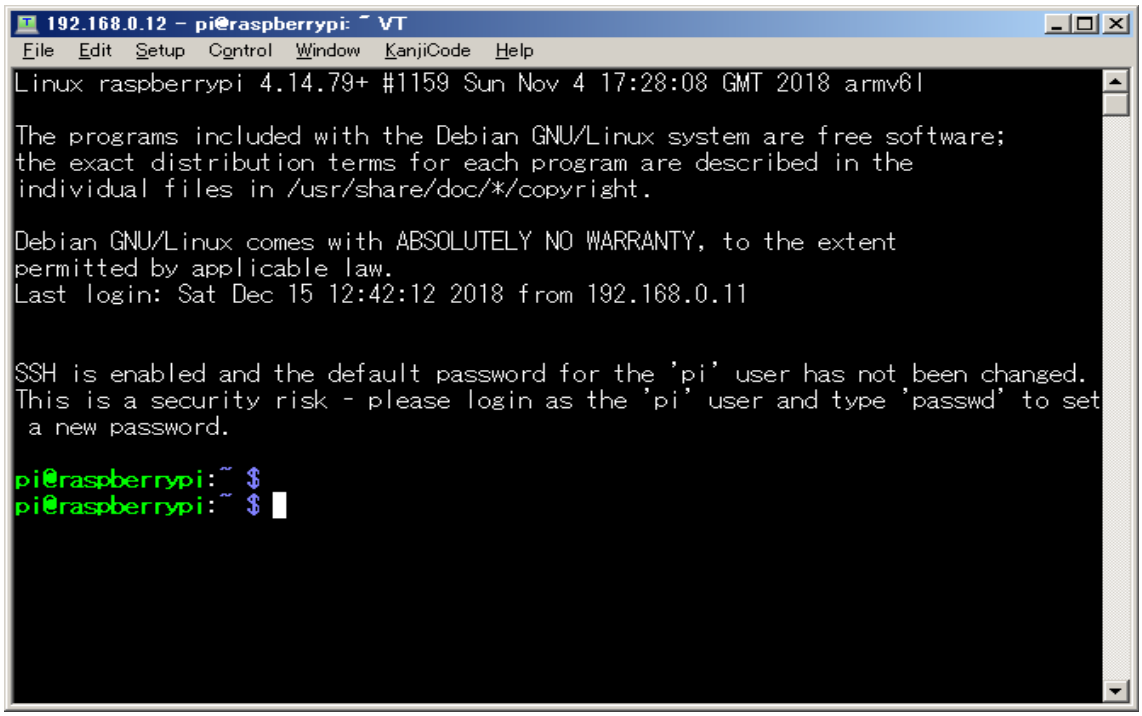
末端の数字を 10 以上増やしてもエコーが返ってこない場合はネットワークの設定が失敗している可能性があります

エコーが返ってくるアドレスが見つかったら、そのアドレスを TeraTerm の **Host** に入れ、OK ボタンを押します

SSH 設定が正常に行われている場合は、ID(user name)と Password(Passphrase)の入力を求められます



Raspberry PI の初期デフォルト ID と Password は **ID:pi Pass:raspberry** です。セキュリティ上問題がありますので、全設定終了後はパスワードの変更を行ってください



ログイン直後の画面

まず最初にやることは、Raspbian の各種モジュールの最新版へのアップデートです。Linux のモジュール周りは日々アップデートを繰り返しているので、セキュリティや機能の強化やバグの退治などが行われています

打ち込むコマンドは~ $ から始まる型で記述していきます。実際に打ち込むときは~ $ の後に続く文字列を入力してください
例：~ $ sudo apt-get と書かれていたら、sudo apt-get と読み替えて入力する

SSH コンソール画面上で
~ $ sudo apt-get -y update
↓終わったら次↓
~ $ sudo apt-get -y upgrade　※アップデート内容によってはかなりの時間がかかります
↓終わったら次↓
~ $ sudo apt-get -y dist-upgrade
↓終わったら再起動↓
~ $ sudo reboot　※OS の再起動、ここで SSH の通信が途切れる(ウインドウが強制的に閉じる)
↓
数分待って再び SSH で再接続(DHCP で割り当てられる IP アドレスが再び変わることはあまり無いので前ので OK)

RaSCSI のディスクイメージを置く場所を決めましょう

基本的にどこでもいいのですが、変なところに置くとファイルシステム権限の関係で色々と面倒くさいことになるので、素直にユーザーディレクトリである/home/pi の下に作ります

現在の位置からユーザーディレクトリに移動します
~ $ cd /home/pi/

移動したら、念のため本当にそのディレクトリにいるかを確認します
~ $ pwd

これですぐ下に /home/pi という文字列が表示されれば OK です

ディスクイメージを置くディレクトリを作ります、好きな名前を付けてください。ここでは例として rasimg というディレクトリ名で進めていきます。この名前は最後の最後まで出てきますので、必ず覚えておいてください

ディレクトリを作るコマンドは mkdir です(MS-DOS でいう MD コマンド)
~ $ mkdir rasimg

ちゃんとディレクトリが作れたか確認をするために、ファイルリストコマンドを入力して確認をします
リストを表示するコマンドは ls です(MS-DOS でいう DIR コマンド)　他にもオプションがありますがここでは省略します
~ $ ls

これですぐ下に rasimg という文字列が表示されれば OK です

ここで一旦ユーザーディレクトリからは離れます

他のマシンとのファイル共有　～samba の設定～

Raspberry PI と他のマシンとでファイルのやり取りをする方法は大まかに
・FTP を使う方法
OS の壁が無い、TCP/IP プロトコルスタックを入れられるのならば MS-DOS からでも接続可能
ただし接続に各 OS に対応した FTP クライアントが必要

・samba を使う方法
Windows マシンとの接続に限れば最も設定が少ない方法
他の Linux マシンや Mac で接続する場合は、ちょっと面倒な設定が必要

今回は後者で説明をしていきます

samba の導入については Linux を使う上での基本中の基本の設定なので、検索すると数多くの解説サイトがヒットします
ディストリビューション(Linux の派生)によっては、同じモジュールでも設定が微妙にローカルな物もある中、samba は定番のせいか、どのディストリビューションでも同じ記述方法でヒットします(もちろんインストールコマンドなどは違いますが)

とりあえず、Raspberry PI への samba の導入については、Raspberry PI＋samba で検索して一番上に来たこのサイトを参考にして説明をしていきます
https://qiita.com/ARBALEST000/items/78f459567e1e90de99e5
まず、システムに samba をインストールします

~ $ sudo apt-get install -y samba

インストール終了までちょっと時間がかかります、しばらくお待ちください

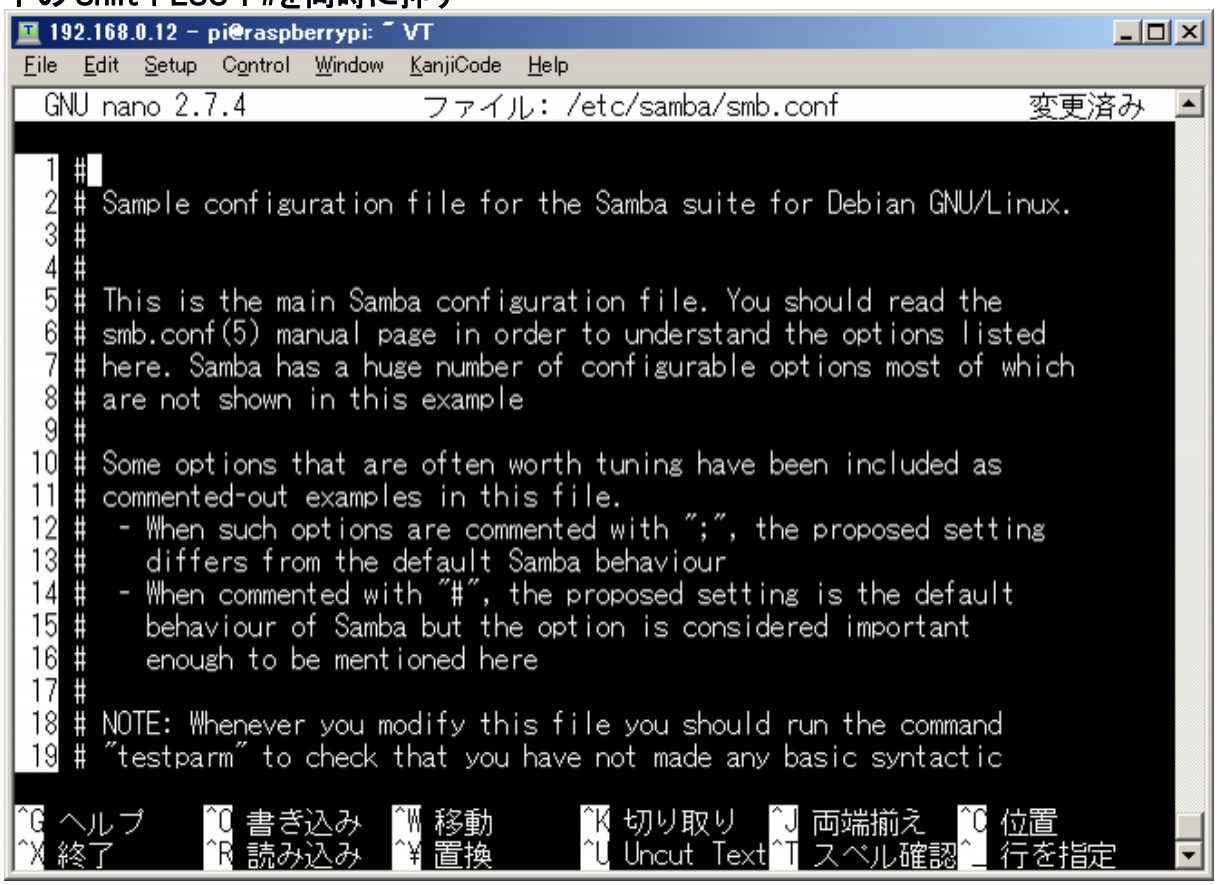
インストールが終わったら、設定ファイルの変更を行います

~ $ sudo nano /etc/samba/smb.conf

※テキストエディタに関しては色々な「派閥」があるようですが、ここでは主に nano を使用していきます

まず、説明を分かりやすくするために nano エディタに行番号を振ります

キーボードの Shift＋ESC＋#を同時に押す



こんな風に行番号が振られます

上記、samba 解説サイトにある「[global]に以下を追記します」をアレンジします

下にスクロールしていくと、20 行台に[global]という文字列が見つかります

global が見つかったらその部分をコピー＆ペーストして、自分のネットワーク環境に合わせて書き換えます

```
interfaces 192.168.0. 127.0.0.1/8 wlan0
bind interfaces only = yes
hosts allow = 192.168.0. fe80::/10
security = user
```

interfaces ＝ 現在、SSH で繋がっているIPアドレスの左 3 アドレス(192.168.0.)
※127.0.0.1 は書き換えないでください
wlan0 ＝ 無線 LAN 接続指定、有線 LAN 接続をしている場合は eth0
hosts allow ＝ これも SSH で繋がっている IP アドレスの左 3 アドレス(192.168.0.)

何をしている部分かというと、指定のIPアドレス以外からの接続を拒否するセキュリティの部分です。この指定を間違えると設定しても接続できなくなります

最後の行にディレクトリの共有設定を入れます
これも解説サイトの部分をコピー＆ペーストしてアレンジします

```
[RaSCSI]
  comment = RaSCSI
  path = /home/pi/
  public = yes
  read only = no
  browsable = yes
  force user = pi
```

[RaSCSI]は Windows の「ファイル共有」で見える共有名
comment はそのままコメントです、好きな識別名を付けてください
path は共有したいディレクトリ、この例ではユーザーディレクトリ/home/pi 以下全てを共有しています

以上、設定が終わったら **CTRL＋X** で保存と終了を行います

samba を再起動して設定を有効にします

```
~ $ sudo systemctl restart smbd
~ $ sudo systemctl restart nmbd
```

ufw のインストールと設定（任意）まわりは必要に応じて設定してください

Windows マシンから、ネットワーク→コンピューターにて「RASPBERRYPI」というコンピューター名、そのコンピューターを開くと「RaSCSI」という共有フォルダ名が確認できれば samba の導入は成功です。以後、Raspberry PI のユーザーホームディレクトリ「/home/pi」は、Windows の論理ドライブとして紐付けされ、ネットワークマウントすることで、ネットワークドライブとして意識することなく使うことが出来ます

なお、コンピューター名(RASPBERRYPI)を変更したい場合は、システム自体のホスト名、共有フォルダ名(RaSCSI)を変更したい場合は、[RaSCSI]の部分を変更してください。この辺のアレンジは samba の解説サイトで勉強してください

おまけ？として、samba を導入すると、今まで IP アドレス直叩きで対応していた RaspberryPI への接続が、DNS で引けるようになります。以後 SSH 接続をする場合はホスト名「raspberrypi」と入力してあげれば、DHCP による IP アドレスの振り直しにも影響されること無く接続や容易になります

ここで一旦 Raspberry PI の電源を落とします

```
~ $ sudo shutdown -h now
```

ここで SSH との接続が切れ、ウィンドウが強制的に閉じられるかエラーメッセージが表示されます

1 分くらいしたら Raspberry PI から電源供給用の microUSB ケーブルを取り外し、完全に電源を断ってください

～3章　RaSCSI プログラムのテスト導入～

ようやく RaSCSI プログラム導入に入ります

最初は RaSCSI アダプタの動作確認と、ハードディスクイメージを試しに作ってコンピューター側で実際に認識・動作するかという確認テストを兼ねた仮構築を行います

まず、Raspberry PI と RaSCSI アダプタを接続し、SCSI ケーブルをコンピューターに接続します。この際、Raspberry PI 側の電源供給手段は、コンピューター側のサービスコンセント等ではなく、必ず常時給電されているコンセントから取ってください

この時点では、コンピューター側の電源はまだ入れないでください

まず最初に Raspberry PI 側の電源を入れ、SSH ログインで操作できる状態にしてください

RaSCSI の公式サイト(GIMONS DEVELOPER WORKS)へ Web ブラウザで移動してください
※http://www.geocities.jp/kugimoto0715/rascsi/　または「RaSCSI」で検索して一番目の検索結果から

ずっと下にスクロールしていくと、ダウンロードという項目が現れます

ここで大切なのは**「ファイルを Web ブラウザ側でダウンロードしない」**ということです、ファイル置き場のリンク情報のみコピーをしてください、後で必要になります

Raspberry PI の SSH ウインドウ側に移ります。ここでユーザーディレクトリ(/home/pi)の下に、RaSCSI の圧縮ファイルを展開するディレクトリを作ります

2018 年 12 月現在のバージョンは 1.34 なので、仮に **rascsi134** とします

~ $ **mkdir /home/pi/rascsi134**
~ $ **cd /home/pi/rascsi134**

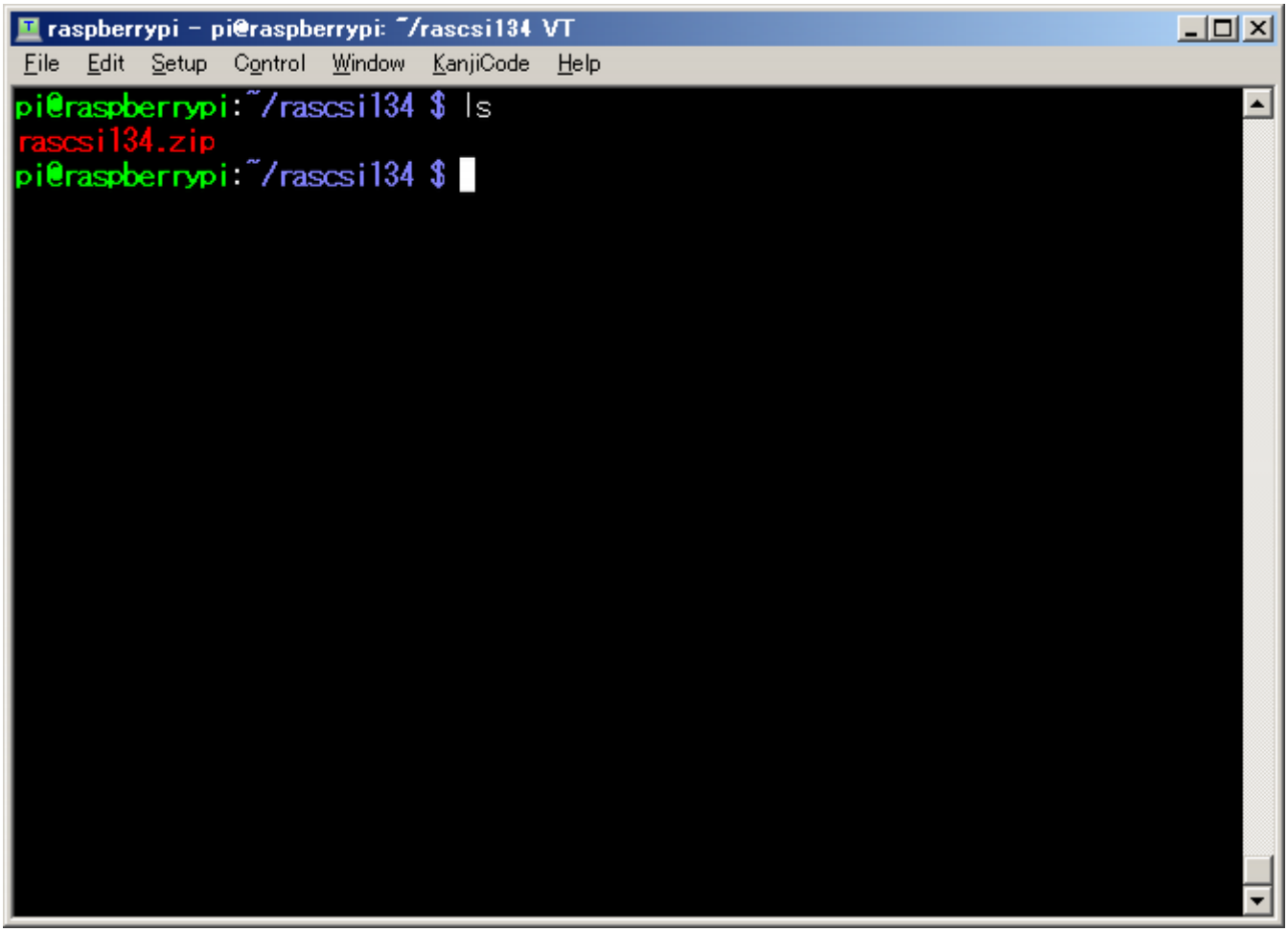
pwd コマンドで、ちゃんと現在地が/home/pi/rascsi134 になっているか確認をします

確認が出来たら、そこに RaSCSI の圧縮ファイルをダウンロードします。ダウンロードコマンドは **wget** コマンドです

SSH ウインドウ上で最初に **wget** と打ち込み、スペースを 1 つ空け、先ほどの RaSCSI ファイルのダウンロードリンクをペーストします(TeraTerm の場合は、右クリック or Alt＋V)

~ $ **wget** http://www.geocities.jp/kugimoto0715/rascsi/rascsi134.zip
※2018 年 12 月現在のURLです。必ず公式サイトからのアドレス取得をお願いします

ls コマンドでちゃんとファイルがダウンロードできたか確認をします



続いてこのダウンロードしたファイルの展開を行います

~ $ **unzip rascsi134.zip**

SSH ウィンドウ上にファイル展開情報が表示されます

展開が終わったら、ls コマンドで直下に

rascsi134 rascsi134.zip

というディレクトリと圧縮ファイルが確認できると思います

何故、rascsi134 というディレクトリが自動的に作られるのに、先に置き場としてのディレクトリを作らないといけないのかというと、たまたまこのバージョン 1.34 はディレクトリ付きで圧縮されていますが、今後のバージョンではその保障がないのと、全作業が終わった後に圧縮ファイルごと「掃除」するのが楽になるという点があげられます(残しておいてもいいです)

さて、ディレクトリ「rascsi134」の中には、メインとして使うバイナリのほか、ドキュメントやソースファイル、X68k で使えるドライバー類などが収められていますが、一部を除いてセットアップには必要の無いものなので、一気に目的のディレクトリまで移動します

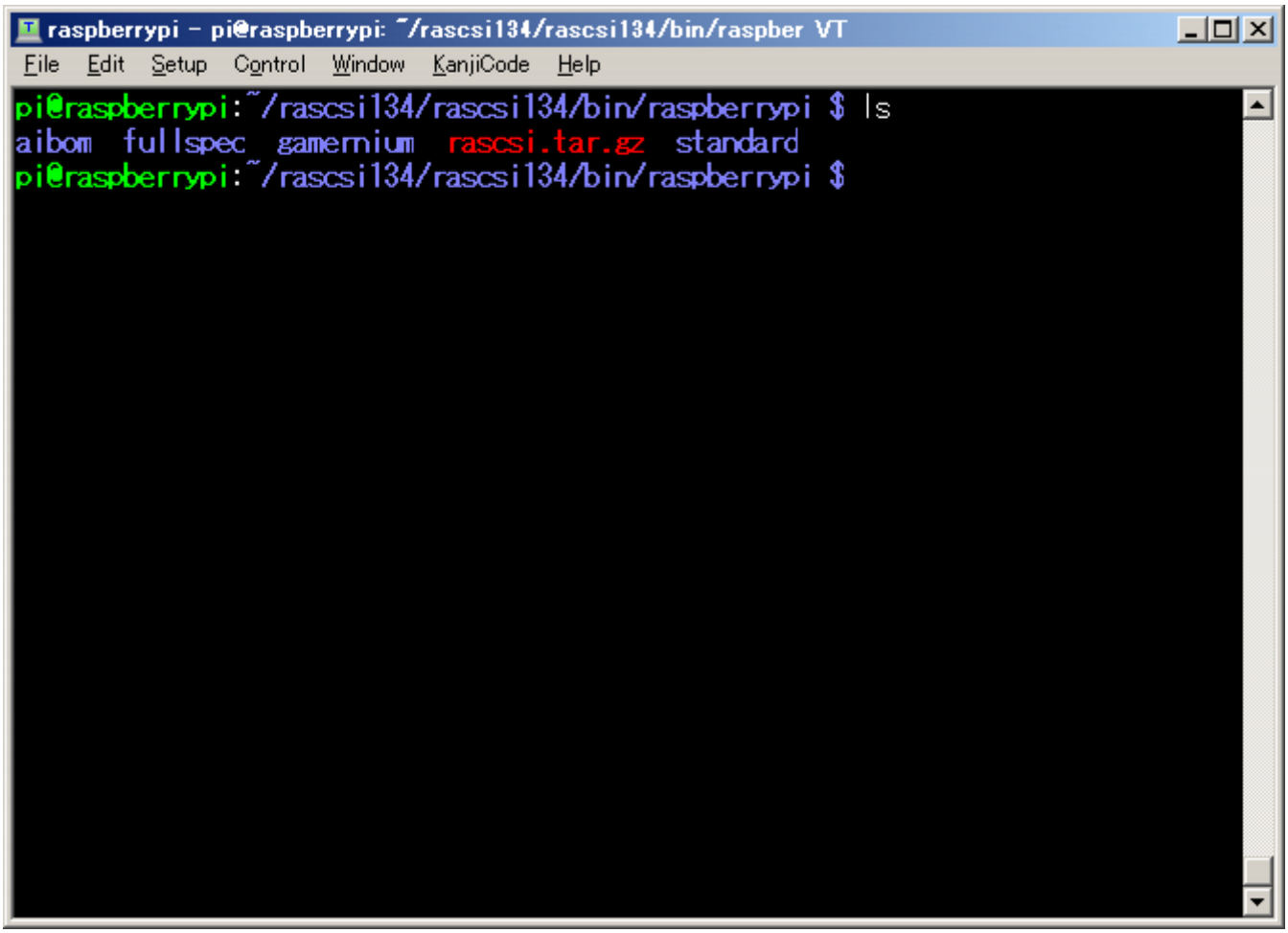
~ $ cd rascsi134/bin/raspberrypi

ここで ls コマンドにて確認すると rascsi.tar.gz というファイルが確認できると思います。このファイルが Raspberry PI で使える RaSCSI バイナリの本体です、今度はこのファイルを展開します

~ $ tar xzvf rascsi.tar.gz

ZIP の展開時と同様、さまざまなファイルの展開情報が表示されます

ここで再び ls コマンドでファイルリストを確認すると、以下のようになっていると思います(一度画面を消去しています)



ここでいよいよ、RaSCSI アダプタの「〇〇版」の知識が生きてきます

- 「GAMERnium 版」の場合は[gamernium]を
- 「あいぼむ版」の場合は[aibom]を
- 「スタンダード版」の場合は[standard]を
- 「フルスペック版」の場合は[fullspec]を

それぞれ選んで cd コマンドでディレクトリを変更してください

ちなみに冒頭で説明したとおり、各ディレクトリの中に入っているファイル群は全て同じファイル名なので、表面上からは識別することが出来ません(コアプログラムを実行すれば分かりますが)

なので、もしバイナリが違う複数の種類の RaSCSI アダプタで共用しようとすると混乱の元になるので、イメージファイルなどの共有は出来なくなりますが、microSD カードの物理的レベルで分けたほうが効率的です

各ディレクトリに移動したら、中に入っているファイルを特定の場所にコピーします

中には 4 つのファイルと 1 つのディレクトリ(rpi1orZero)がありますが、そのうち必要なのは
[rascsi][rasctl][rasdump]の 3 ファイルになります(黄緑色で示されていると思います)
この 3 ファイルをコピーします

さて、これらファイルをコピーする場所ですが、これでもLinuxの使用者によって「派閥」みたいなものがあるようですが、私は「**/usr/local/bin**」を推しています

そんなわけで、ファイルを/usr/local/binへコピーします

~ $ **sudo cp rascsi rasctl rasdump /usr/local/bin**

~ $ **ls /usr/local/bin と入力**して、ちゃんと3ファイルがコピーできたか確認します

確認が出来たらユーザーディレクトリまで移動します(/home/pi)

ここでようやくSSH接続編の最初にやった「rasimg」ディレクトリが登場します

まずrasimgディレクトリに移動します

~ $ **cd rasimg**

このディレクトリ直下にテスト用のハードディスク空イメージを作成します

PC-98やX68kエミュレータのディスクイメージ作成機能で作る方法もありますが、Linuxの古来からの由緒あるディスク操作コマンドである**ddコマンド**で作ります

~ $ **dd if=/dev/zero of=scsiimg0.hds bs=1M count=40**

scsiimg0.hdsというのがイメージファイル名
1Mというのは容量単位
40というのは作成する容量を現します

つまり、今回は40Mの空ディスクイメージを作るということです
ですので、100Mのイメージを作りたい場合は100を、1GBのイメージを作りたい場合は1024を指定してください

なお、bsの1Mはメガバイト(10^6＝1,000,000バイト)ではなく、メビバイト(2^20＝1,048,576バイト)の容量になります

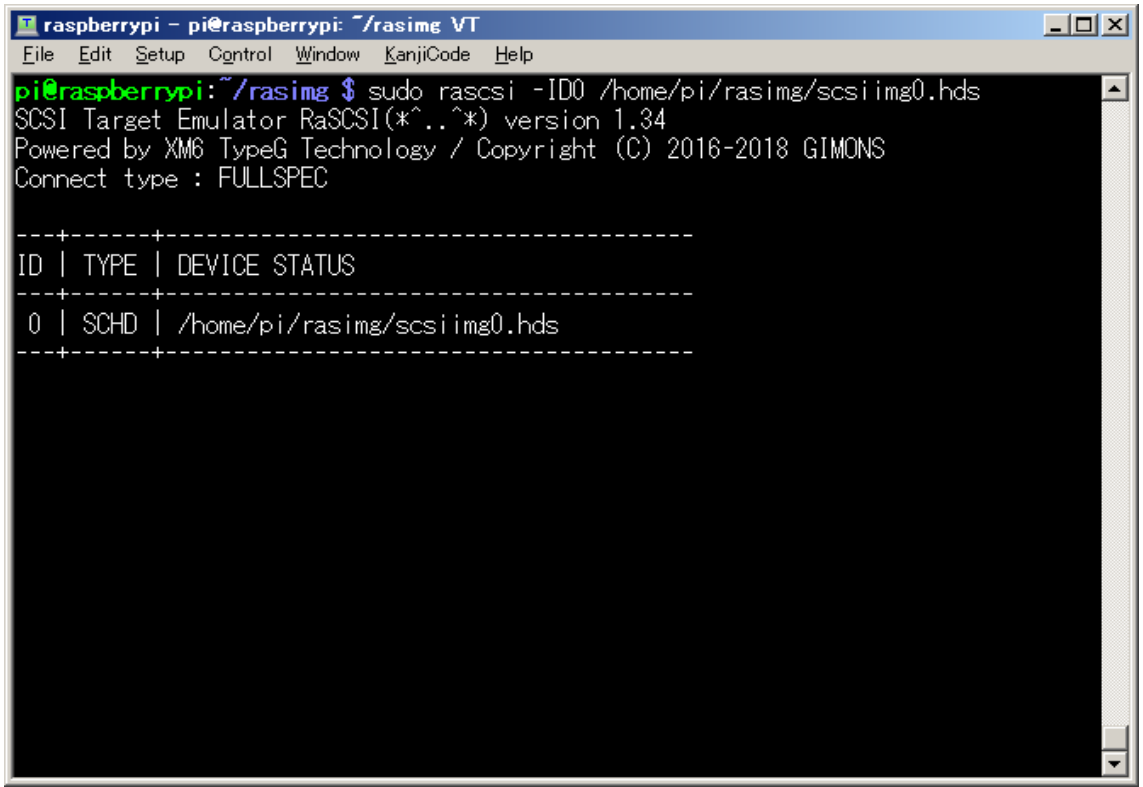
この辺はちょっと容量指定やディスクイメージの拡張子回りでトリッキーな方法を
要求される部分があるので、最後のほうにて補足解説をしたいと思います

lsコマンドでscsiimg0.hdsというファイルが確認できればOKです

RaSCSIプログラムのテスト実行

ここで、先ほど作った空ディスクイメージファイルをrascsiプログラムに割り当てます
~ $ **sudo rascsi -ID0 /home/pi/rasimg/scsiimg0.hds**

書式は
rascsi → RaSCSIのコアプログラム名　ID0 → 割り当てるSCSI-ID(今回は0) /home/～ディスクイメージへのフルパス



この画面で他の入力などを受け付けない一時停止の画面になったら、コンピューターの電源を入れます

コンピューター側の SCSI アダプタでデバイススキャンの際、ID0 にディスクが認識されるか、スキャン画面が出ない場合はフロッピーディスクでシステムブートし、ハードディスクのフォーマットコマンドでハードディスクが認識されればテスト成功です

SCSI の認識テストが成功したらコンピューターの電源を落とします。そして SSH ウインドウ上で **CTRL＋C** を押して、RaSCSI の動作を一旦終了させます

RaSCSI カーネルモジュールのコンパイルと組み込み

RaSCSI カーネルモジュールとは、RaSCSI の GPIO の一部機能をよりシステムに近い低レベル領域で実行させ、CPU 負荷の低減や、動作の安定性を向上させる間接モジュールです

システムにより近い低レベル領域で機能するため、Raspberry PI の OS、すなわち Raspbin(Linux)のカーネルに強く依存し、基本的にカーネルバージョンが異なる間のカーネルモジュールの融通が出来ず、現在稼働中のシステムに合わせて都度ソースファイルからコンパイルしなおして組み込む必要があります

RaSCSI はカーネルモジュールを入れなくても案外動くことが多いのですが、X68k の SASI や SCSI の動作関係で入れないと動作が不安定になることが多いようです。なので X68k で使う場合は導入が必須、他の機種で使う場合でも動作不安定な要素を残したくないのであれば入れることを推奨いたします

まず、RaSCSI のセットアップを冒頭から行っている場合は必要ありませんが、日付をおいて別途カーネルモジュールだけ組込もうとしている場合は作業前に現在の Raspbian を最新版にアップデートします(上記のOSインストール直後編参照)

```
~ $ sudo apt-get -y update
~ $ sudo apt-get -y upgrade
~ $ sudo reboot
```

続いて Linux のカーネルヘッダーを取得します

```
~ $ sudo apt-get install raspberrypi-kernel-headers
```

※ダウンロード～展開～インストールまで少々時間がかかります

続いて RaSCSI のカーネルモジュールのソースが置いてあるディレクトリまで移動します

```
~ $ cd /home/pi/rascsi134/rascsi134/src/raspberrypi/kernelmodule
```

※このマニュアルの手順どおりに作業を行っている場合のディレクトリパスです。カーネルモジュールのコンパイル～組み込みからの場合は(RaSCSI ファイルの展開場所)/src/raspberrypi/kernelmodule で移動してください

カーネルモジュールのソースが収められているディレクトリで ls コマンドを入力すると、Makerfile／rascsidrv.c という 2 つのファイルが確認できると思います

rascsidrv.c ソースファイルの書き換え

じつは RaSCSI の組み込みに比較的慣れてきた熟練者でも非常に陥りやすい落とし穴トラップです。このソースファイル(rascsi.drv.c)にも〇〇版の RaSCSI アダプタごとの「設定」が隠れています。これをやらずにコンパイルをして組み込んだカーネルモジュールが原因でコンピューター側で SCSI として認識されずに悩むパターンが見られます

前項の RaSCSI テスト実行は RaSCSI アダプタとしても、〇〇版のバイナリプログラムとしても、ちゃんと正常に動くかを確認する本当の意味での「テスト実行」になります

エディタで rascsidrv.c を開きます

```
~ $ nano rascsidrv.c
```

あと、分かりやすくするため行番号(Shift＋ESC＋#)もつけます

(次ページのスクリーンショット参照)

行番号の 41～44 行にわたり、各 RaSCSI アダプタに応じた設定に書き換えます

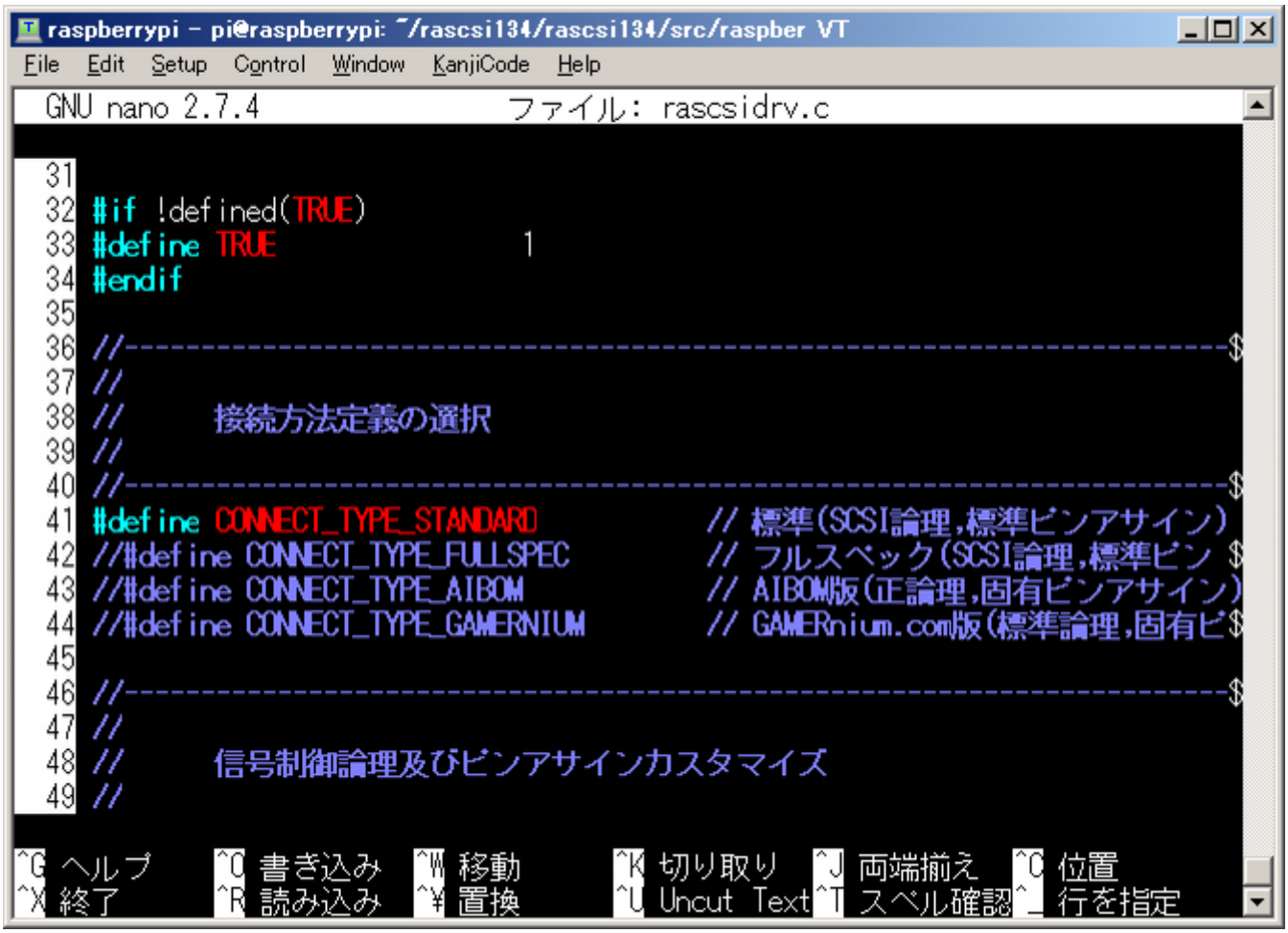
例えば GAMERnium 版のアダプタを使用している場合

最初はスタンダード版が有効になっていますが、これを//でコメントアウトし、代わりに GAMERnium 版の行の//を削除して有効にします

```
#define CONNECT_TYPE_STANDARD
//#define CONNECT_TYPE_FULLSPEC
//#define CONNECT_TYPE_AIBOM
//#define CONNECT_TYPE_GAMERNIUM
```

↑コレを↓のように書き換える

```
//#define CONNECT_TYPE_STANDARD
//#define CONNECT_TYPE_FULLSPEC
//#define CONNECT_TYPE_AIBOM
#define CONNECT_TYPE_GAMERNIUM
```



このように、現在使用している RaSCSI アダプタに応じた設定書き換えを行ってください

書き換えが終わったら、CTRL+X でセーブと終了をします

引き続き、カーネルモジュールのコンパイルを行います

```
~ $ make
```

コンパイルした RaSCSI のカーネルモジュール(rascsidrv.ko)のインストール

```
~ $ sudo insmod rascsidrv.ko
```

ちゃんとカーネルモジュールが組み込まれたか確認

```
~ $ dmesg | grep "RaSCSI GPIO Driver"
```

RaSCSI GPIO Driver Loaded(xxxxxx)
※xxxxxx は組み込まれたモジュールタイプ(STANDARD or FULLSPEC or aibom or GAMERnium.com)

この状態で一旦ハードディスクイメージが置いてあるディレクトリに戻り、再び

```
~ $ sudo rascsi -ID0 /home/pi/rasimg/scsiimg0.hds
```

とコマンドを入力してみてください

Connect type : xxxxxx の下に **Activated RaSCSI GPIO Driver** という 1 行が増えていると思います。この状態でふたたびコンピューターの電源を入れ、コンピューター側から SCSI デバイスとして認識～可能ならブートまで可能か確認します。確認が出来たら再び電源を落として、RaSCSI プログラムを CTRL+C で終了させます

さて、結構大変なカーネルモジュールのコンパイル～組込みまでの作業ですが、ここで大変な事実が
「Raspberry PI を再起動または電源を落として入れなおすとカーネルモジュールの組込み設定が消える」です
ここらへんの解決方法は次章の「RaSCSI の自動実行設定」にて説明をします

～4章　RaSCSI の自動実行設定～

「レトロPCのストレージ事情に革命を起こした RaSCSI」ですが、使えるまでの設定の難解さもさることながら、仮想ストレージとして使うために、いちいち SSH で接続してカーネルモジュールや RaSCSI プログラムを手動で実行していては、いくら超便利な存在だとしてもやっていられません

そんなわけで、Raspberry PI の起動時に自動実行できるところは自動実行の設定にしてしまいます

まず、カーネルモジュールのシステムへの組込みです

前回コンパイルした RaSCSI のカーネルモジュール(rascsidrv.ko)を、Raspberry PI のカーネルモジュールが多数収められているところにコピーします

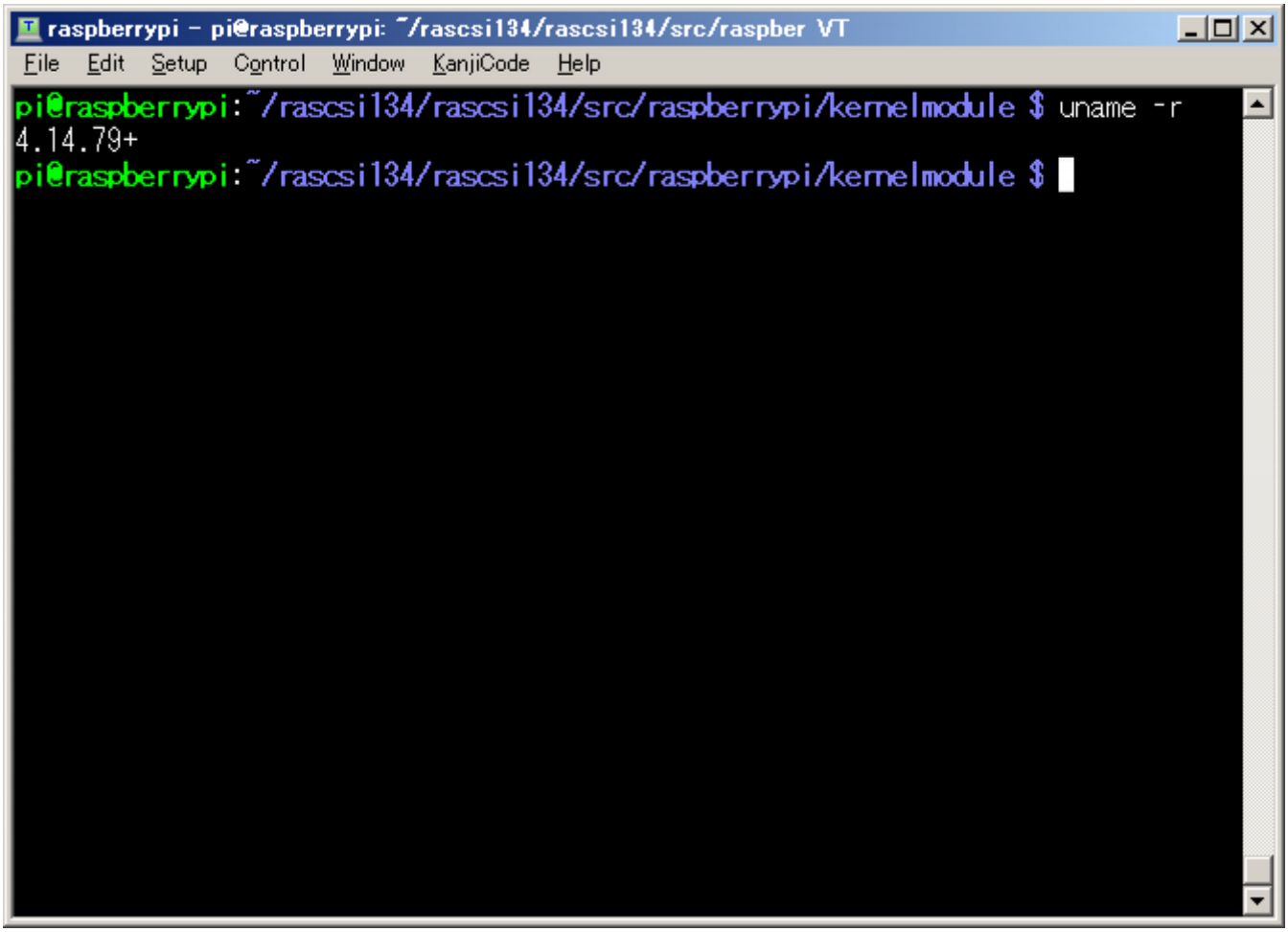
まずは、RaSCSI のカーネルモジュールが置いてあるディレクトリまで移動

~ $ **cd /home/pi/rascsi134/rascsi134/src/raspberrypi/kernelmodule**

RaspberryPI のカーネルモジュールが置いてあるディレクトリはカーネルバージョンによって違うため、まず動作中の Raspbian のカーネルバージョンを割り出します

~ $ **uname -r**

コマンドの下に 4.xx.xx　みたいな数字が表示されますが、これが現在動作中の Raspbian のカーネルバージョンです
（下のスクリーンショットでは 4.14.79+ってなっていますね）



これをふまえて、RaSCSI のカーネルモジュールをコピーします

~ $ **sudo cp rascsidrv.ko /lib/modules/(カーネルバージョン)/**

※上のスクリーンショットに倣うならば sudo cp rascsidrv.ko /lib/modules/4.14.79+/

その後、先ほど RaSCSI のカーネルモジュールをコピーしたディレクトリを確認します

~ $ **ls /lib/modules/(カーネルバージョン)/**

ファイル群の中に rascsidrv.ko が確認できれば OK です

続いて、カーネルモジュールを自動組込する設定ファイルを nano エディタで書き換えます
~ $ **sudo nano /etc/modules**

開いたファイルの末端行に **rascsidrv** とだけ書き込み、CTRL-X でセーブと終了をします

同時に他のカーネルモジュール間との依存関係も解消します
~ $ **sudo depmod -a**

ここで一旦、RaSCSI カーネルモジュールが自動的に組み込まれるか確認するために Raspberry PI を再起動します
~ $ **sudo reboot**
(再起動終了まで少し待つ→SSH で再ログイン)

~ $ **dmesg | grep "RaSCSI GPIO Driver"**
RaSCSI GPIO Driver Loaded(xxxxxx)

何度再起動して上のコマンドを入力しても、その下のようなメッセージが出てこれば自動組込み OK です

カーネルモジュールの自動組込みが確認できたら、ハードディスクイメージの自動マウント部分を設定します。nano エディタで/**etc/systemd/system/**ディレクトリに **rascsi.service** というファイルを新規で作成します

~ $ **sudo nano /etc/systemd/system/rascsi.service**

そこに、以下の設定スクリプトをそのままコピー＆ペーストした後、CTRL+X でセーブと終了をします

```
[Unit]
Description=RaSCSI_Service
After=syslog.target

[Service]
Type=simple
ExecStart=/usr/bin/sudo /home/pi/rasimg/rasmount.sh
TimeoutStopSec=5
StandardOutput=null
Restart=no

[Install]
WantedBy = multi-user.target
```

各オプションがどういう意味かは・・・正直自分もあんまり理解しておりません(ダメじゃん)

この中で気をつけるのは **ExecStart オプション**
パスは RaSCSI のディスクイメージが置いてあるディレクトリ(/home/pi/rasimg)に繋がっています

ExecStart オプションは「システムの起動時に＝の先に書いてあるコマンドを実行する」というものなので、ここに直接 RaSCSI のディスクマウントコマンドを書いてもいいのですが、場所が/**etc/systemd/system/**という関係上、起動時に自動マウントされるハードディスクイメージ等を変更したい時、サービススクリプトの置き場所をよく忘れるため、実際のディスクイメージをマウントする部分は RaSCSI のディスクイメージが置いてある場所に一緒にしておいたほうがいい、ということで

上記スクリプトファイルが作成できたら、サービス自動起動設定をします

~ $ **sudo systemctl enable rascsi**

これでシステム起動時に毎回上記のスクリプトが自動実行されます

次に上記スクリプトの ExecStart の部分、すなわち RaSCSI のディスクイメージが置いてある場所に rasmount.sh というシェルスクリプトを作ります。ディスクイメージが置いてあるディレクトリに移動します
~ $ **cd /home/pi/rasimg**

nano エディタで rasmount.sh というファイルを新規に作成します
~ $ **nano rasmount.sh**

ここでも上記のサービス起動スクリプトと同様、以下の設定スクリプトをそのままコピー＆ペーストした後
CTRL+X でセーブと終了をします

```
#!/bin/sh
hdir=/home/pi/rasimg/
rascsi -ID0 ${hdir}scsiimg0.hds -ID6 bridge
```

スクリプトの説明
#!/bin/sh
このファイルはシェルスクリプトであるという指定

hdir=/home/pi/rasimg/
RaSCSI ディスクイメージへのファイルはフルパスで指定しないといけないが、全ての SCSI-ID に対し
フルパスで書いていると長くなるので、共通部分は変数化して短縮する

rascsi -ID0 ${hdir}scsiimg0.hds -ID6 bridge
RaSCSI コアプログラム実行部分。全ての ID-イメージファイル名を 1 行で記述する。細かいオプションは後述

最後に rasmount.sh ファイルに「実行権限」をつけて終了です

```
~ $ chmod 755 rasmount.sh
```

RaSCSI(自動実行モード)の動作確認

全ての設定が終わりましたので最終動作確認をします。まず Raspberry PI を一度シャットダウンします

```
~ $ sudo shutdown -h now
```

Raspberry PI の LED が点滅しなくなったら、AC アダプタか microUSB コネクタを抜いて一度完全に電源を抜きます

手順どおりに設定をしている場合は、RaSCSI アダプタはコンピューターに挿さったままになっていると思いますが、もし取り外している場合は、再び SCSI ボードに接続してください

一息ついて、再び Raspberry PI に電源を供給します。コンピューターのサービスコンセントからでも、常時給電されている別のコンセントからでもかまいません

Raspberry PI の microSD カードのアクセス LED が落ち着いたらコンピューターの電源を入れます。SCSI ボードのデバイススキャンまたはフロッピー起動によるフォーマットコマンドでハードディスクが無事認識されていればRaSCSIの自動起動は成功です、おめでとうございます！

～終章　終わりに～

こんな長くてまとまりが無いドキュメントに最後までお付き合いいただきありがとうございます

今回 RaSCSI アダプタの導入部分から、自動実行による SCSI ディスクイメージの自動マウントまで書きましたが、これらは RaSCSI のほんの「導入部分」に過ぎません。実際にはまだまだやることが残っています

ただ実際問題、物珍しさで結構いいお値段がする RaSCSI アダプタを買ってみたはいいが、実際に設定できて活用できている人はそう多くないと思います。RaSCSI というテクノロジーは大変素晴らしいものですが、同時に Linux に関するある程度の知識も要求されるわけで、古いコンピューターを使っているからといって、古今の情報知識万能というワケではないのです

古いコンピューターをハンダゴテで修理できたり、プログラミングできたりする人もいれば、それこそフロッピーディスクやカセットテープをロードして当時のゲームをプレイするだけのユーザーもいます、昔からコンピューターを使っている＝詳しい、というわけではないのです

冒頭にも書きましたが、ハードウェアとしての RaSCSI アダプタは一種の到達点を迎えました。これ以上の進化は革新的な機能がソフトウェア側で実装されない限り、基本型をベースに付加価値をつけるか、機能を限定しつつ部品点数を減らしてコストダウンを図る方向にシフトしていくものと思われます

しかし、ソフトウェア面の進化は RaSCSI が軌道に乗った 2017 年中ごろとほとんど変化がありません。ここでいう「進化」とは、RaSCSI プログラムのことを言うのではなく、RaSCSI というテクノロジーを初心者にも出来る限りとっつき易くするという進化です。RaSCSI の初期導入に関しては、岩崎さんのページ(MUSHAGAESHI.COM)がほぼバイブルとなっていますが、やはりどうしても Linux の前提知識があってこその流れになっているので難しいところです

「はじめての RaSCSI 導入マニュアル」は、情報過多の Raspberry PI 初期導入まわりから、RaSCSI で必要な部分のみ抜粋して説明、逆に情報が限定的 or 足りなさ過ぎな部分は「とりあえずこの説明どおりにやれば最低限使えるから！」とカスタマイズ性を犠牲にしてでも直線的な説明で進めています

かなり Raspberry PI 導入初心者にも分かるように書いたつもりですが、正直これでもどこか引っかかると思います。そういう時「ここがよく分からないです」というフィードバックがあれば、より噛み砕いた説明もできるのですが、意外とそういうフィードバックって無いんですよね・・・(涙)

と・に・か・く！RaSCSI はハードウェアの知名度に対して、どうしても柔軟性に欠ける「実際に使えるまでの道」があまりにも細く険しすぎるので、この導入指南書がきっかけとなって、RaSCSI という超便利テクノロジーを広く、出来る限り多くのレトロコンピューターユーザーにお届けできるようになるといいですね

2018 サークル *Project M.P.S*
Write of シエル

おまけ：補足解説いろいろ～Raspberry PI システム編～

「コンピューター」
この単語には操作する側としての Windows マシン、Raspberry PI、レトロコンピューター、全ての意味で共通となっております。前後の文脈で適切に読解してください

「フォルダ」と「ディレクトリ」
本書では、Windows から見たファイル類の入れ物を「フォルダ」、Raspbian から見たファイル類の入れ物を「ディレクトリ」と表記しています。決して気分でブレブレなワケではありません

ホスト名やパスワード変更は raspi-config でも変更できますが、将来別の Linux ディストリビューションを使う可能性も考慮して(?)、あえてコマンドラインから変更を行います

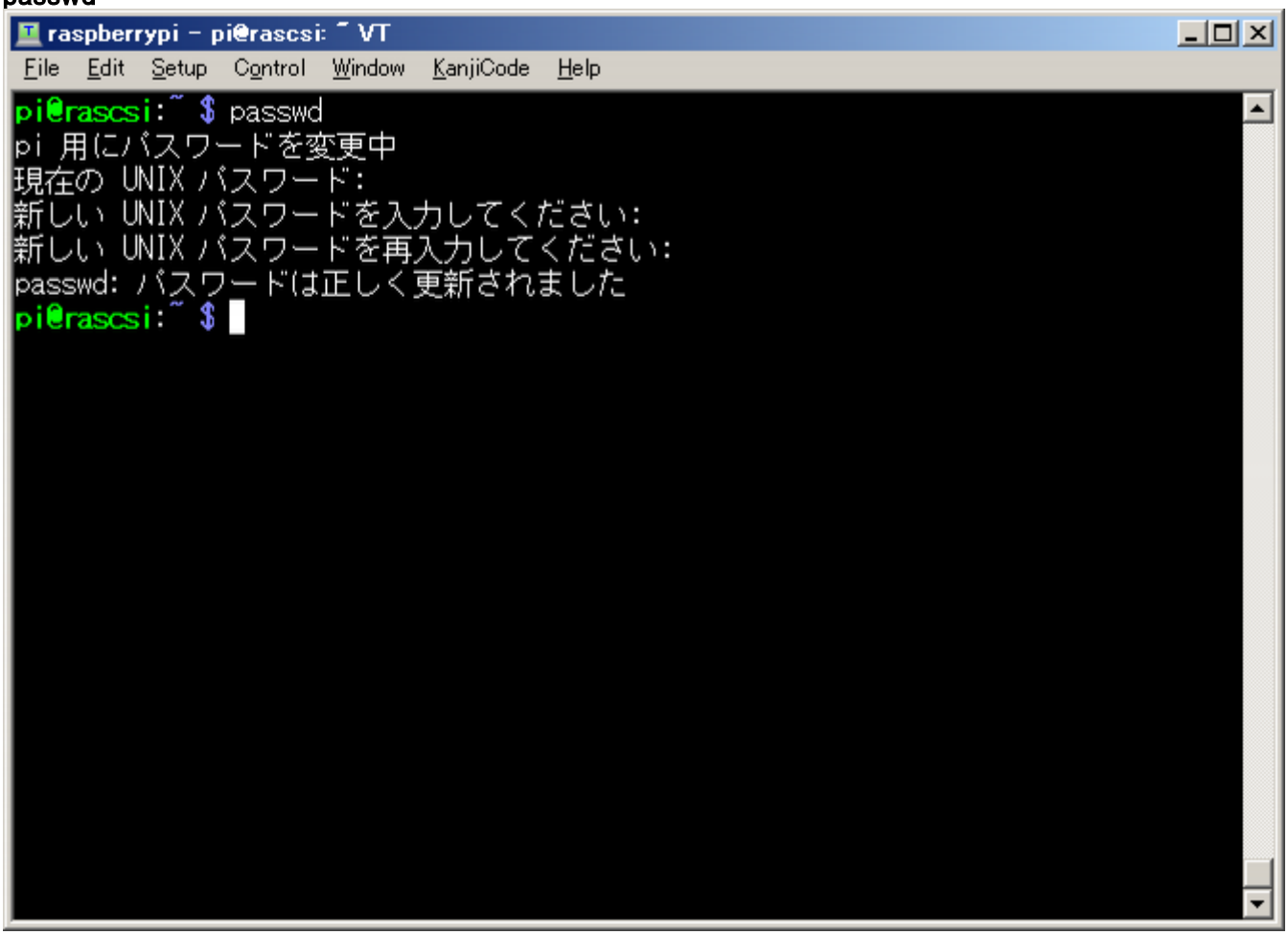
「Rasperry PI のホスト名の変更」
デフォルトのホスト名は「raspberrypi」ですが、複数の Raspberry PI を運用する際に同じ名前だと混乱をしますし、samba の共有名もホスト名に引きずられます、なので必要に応じて変更してください。なお、ホスト名を変更しても RaspberryPI は他の設定ファイルの書き換えなどは必要ありませんが、SSH 接続の際に名前解決にホスト名を使っている場合は、そちらのほうは書き換える必要があります

仮にホスト名を「rascsi」とする場合
~ $ **sudo hostnamectl set-hostname rascsi**

この後、システムを再起動させて設定を反映させます。samba まわりの設定反映は少し時間かかるかもしれません

「Raspberry PI のパスワード変更」
Raspberry PI への raspbian インストールの時にも触れましたが、仮にもネットワークに繋がっているデバイスが、デフォルトパスワードのままではセキュリティ的に色々マズいので、パスワードの変更を行います

~ $ **passwd**



当然のことながら「現在の UNIX パスワード」「新しい UNIX パスワード」共々、入力しても入力中の文字は見えません

「RaSCSI と Raspberry PI の終了」
マルチタスクOSという関係上、終了＝電源ブチッ、は色々とマズいので、面倒ですが出来る限りコンピューターの電源を落とした後に、SSH でログインして **sudo shutdown -h now** で Raspberry PI の電源を安全に落としてください。将来的に何らかの対策が取られるかもしれません

おまけ:補足解説いろいろ～RaSCSI 編～

rascsi コアプログラムの起動オプション簡易解説
~ $ **sudo rascsi -h** で説明が出ますが、慣れていないとちょっと分かりづらいので補足

SASI インターフェース搭載機(X68000 初代/ACE/PRO/EXPERT や PC-9801-27 など) ~ $ **sudo rascsi -ID0 scsiimg0.hdf** 拡張子「hdf」、1 台あたり最大 40MB、2 台まで可能(X68k でドライバを使えばまた変わるらしいですが)
SCSI インターフェース搭載機(X68000 SUPER 以降、PC-9801-55**「以外」**の 98 用 SCSI ボード、FM TOWNS など) ~ $ **sudo rascsi -ID0 scsiimg0.hds** 拡張子「hds」、最大容量は SCSI-BIOS に依存、7 台まで可能。RaSCSI の一番標準的な指定オプション
SCSI インターフェース搭載機(PC-9801-55 系インターフェース) ~ $ **sudo rascsi -ID0 scsiimg0.hdn** 拡張子「hdn」、最大容量 400MB、4 台まで可能。俗に言う「NEC チェック」トラップがかかっている機種用
SCSI インターフェース搭載機(Apple Macintosh) ~ $ **sudo rascsi -ID0 scsiimg0.hda** 拡張子「hda」
MO ディスクイメージ(SASI 機の場合指定不可) ~ $ **sudo rascsi -ID0 scsiimg0.mos** 拡張子「mos」、RaSCSI の「イニシエイターモード」搭載機で吸出しをしたイメージ向け、あまり使われることは無いと思う
CD-ROM ISO イメージ(SASI 機の場合指定不可) ~ $ **sudo rascsi -ID0 scsiimg0.iso** 拡張子「iso」、CD-ROM イメージ化ソフト(ImgBurn など)や、Raspberry PI に光学ドライブを繋いで dd コマンドでイメージ化したファイル。CD-DA 部分は使えない
ブリッジオプション(X68000 で真価を発揮する) ~ $ **sudo rascsi -ID6 bridge**

SCSI のデージチェーンをする場合は
~ $ **sudo rascsi -ID0 scsiimg0.hds -ID1 scsiimg1.hds -ID2 moimg.mos -ID3 cdimg.iso -ID6 bridge**
と一列で書く

基本的に rascsi のコアプログラム実行にパスは必要ないが、イメージファイルの指定には絶対パスが必要(自動実行参照)

もう一つ、rasctl というディスクイメージを動的に切り替えられるプログラムがあります。こちらは rascsi コアプログラムの実行中のみ有効なオプションで、再起動や電源を切ると変更情報は失われます

~ $ **rasctl** オプション無しで実行すると、使い方説明が出ます

~ $ **rasctl -l** 現在の仮想ディスクの稼働状況表示

```
---+-------+-----------------------------------------
 ID | TYPE  | DEVICE STATUS
---+-------+-----------------------------------------
  0 | SCHD  | /home/pi/rasimg/scsiimg0.hds
---+-------+-----------------------------------------
```

こんな感じにリストが出ます

空き SCSI-ID に仮想ハードディスクを追加する場合(SCSI-ID 1 の場合)

~ $ **rasctl -i 1 -c attach -t hd -f scsiimg1.hds**

これだと、どうしても入力コマンドが長くなるので省略も可能です

~ $ **rasctl -i1 -ca -th -f scsiimg1.hds**

どちらでも可能ですので、とっつきやすい方法を入力してください
(もっと省略することも可能ですが、わけ分からなくなるのでここまでとします)

MO を追加する場合
~ $ **rasctl -i 1 -c attach -t mo -f scsiimg1.mos** または~ $ **rasctl -i1 -ca -tm -f scsiimg1.mos**
CD を追加する場合
~ $ **rasctl -i 1 -c attach -t cd -f scsiimg1.iso** または~ $ **rasctl -i1 -ca -tc -f scsiimg1.iso**
ブリッジモードを追加する場合
~ $ **rasctl -i 6 -c attach -t bridge** または~ $ **rasctl -i6 -ca -tb**
※こちら方も、ブリッジモード以外ディスクイメージ指定にはフルパス指定が必要です

逆に仮想ディスクを削除する場合は
~ $ rasctl -i 1 -c detatch または~ $ rasctl -i1 -cd

削除する場合は、SCSI-ID とディスクを削除するコマンドだけ指定すれば OK です

なお、MO と CD に関してはメディアを挿入しない「ドライブだけ」のマウントはできないようです。必ず何かしらのイメージファイルをマウントした後、メディアを交換するコマンドが必要です

これらハードディスク系は、Raspberry PI 上でイメージを切り替えたら

「必ずコンピューターをリセットして再び SCSI デバイススキャンを行ってください」

データの不一致が起こり、データ破壊の原因にもなります

MO／CD などのリムーバブルメディア特有のオプション。上記で MO／CD イメージのマウントを行った上で、必要に応じてメディアイジェクト→メディアインサートの手順を踏みます

メディアの挿入(MO)
~ $ rasctl -i 1 -c insert -t mo -f motest.mos
または
~ $ rasctl -i1 -ci -tm -f motest.mos

メディアの挿入(CD)
~ $ rasctl -i 1 -c insert -t cd -f cdtest.iso
または
~ $ rasctl -i1 -ci -tc -f cdtest.iso

メディアのイジェクト(MO/CD)
~ $ rasctl -i 1 -c eject または ~ $ rasctl -i1 -ce

メディアのライトプロテクト(MO)
~ $ rasctl -i 1 -c protect または ~ $ rasctl -i1 -cp
※MO はメディアのインサートとプロテクト同時にはできません、別々にコマンドを実行する必要があります
※プロテクトを解除する場合はもう一度上記のコマンドを入力する必要があります

MO/CD のリムーバブルメディアディスク交換は、必ず MO/CD の SCSI-ID は一致させてください

各種ディスクイメージの作成

上のテストイメージデータ作成では、下記のようにザックリと 40M で作ってしまいましたが、SASI ディスク形式や MO などはサイズが決まっているので、この通りにはできません(逆に SCSI ディスク形式は結構融通が利きます)
~ $ dd if=/dev/zero of=scsiimg0.hds bs=1M count=40

ディスクイメージは従来は XM6 TypeG などの各機種用エミュレータで作るのが定番でしたが、実際にはこれらディスクイメージは実際にはただのベタデータなので、Linux 標準の dd コマンドで作れます

ただし、全て dd で OK というワケではなく、SCSI ボードのディスクパラメータ取得で数値が取得できないなどの「相性」により、この方法で作ったイメージファイルが仮想 SCSI ディスクとして認識されないことがあります。そういう場合には従来どおり、各種エミュレータを使ってディスクイメージ作成して Raspberry PI に転送してください

SASI 形式(10M／20M／40M)

ハードディスクのセクタサイズは一部の例外を除いて 512 バイト／セクタなので、512 バイト×nn 倍でディスクイメージを作ります。欠点はセクタサイズを小さくするとイメージ作成に少し時間がかかること。40MB 程度なら特に問題にはなりませんが、1GBとかの大容量を 512 バイト単位で作ろうとすると大変な時間がかかります

RaSCSI のディスクイメージが収められているディレクトリで、以下のコマンドをファイル名の部分だけアレンジして
コピー&ペーストしてください

SASI 10MB(10,441,728 バイト)
~ $ dd if=/dev/zero of=sasi10m.hdf bs=512 count=20394

SASI 20MB(20,748,288 バイト)
~ $ dd if=/dev/zero of=sasi20m.hdf bs=512 count=40524

SASI 40MB(41,496,576 バイト)
~ $ dd if=/dev/zero of=sasi40m.hdf bs=512 count=81048
※SASI-HDD での動作確認環境が無い為、これで本当にディスクとして認識されるかどうかは分かりません

MO 形式(128M／230M／540M／640M)

MO はハードウェアセクタ方式で 640MB を除き 512 バイト／セクタと決まっています(640MB のみ 2048 バイト／セクタ)

元々 RaSCSI での MO サポートは、イニシエイターモードサポートの RaSCSI アダプタで MO メディアからのイメージ吸出しをしたファイルを取り扱うためにサポートされたようなものなので、わざわざイメージデータを作るという意義があるのかは疑問ですが、一応書いておきます

MO 128MB(127,398,912 バイト)
~ $ **dd if=/dev/zero of=mo128m.mos bs=512 count=248826**

MO 230MB(228,518,400 バイト)
~ $ **dd if=/dev/zero of=mo230m.mos bs=512 count=446325**

MO 540MB(533,248,000 バイト)
~ $ **dd if=/dev/zero of=mo540m.mos bs=512 count=1041500**

MO 640MB(635,600,896 バイト)
~ $ **dd if=/dev/zero of=mo640m.mos bs=2048 count=310352**

各種、容量が結構大きいのでイメージデータ作成に時間がかかります
※こちらは PC-98 にて動作確認が取れました

有志が作成した RaSCSI で使える便利ツール

・RASDRV.SYS
・RASETHER.SYS
本家(GIMONS DEVELOPER WORKS)

X68k で使える、RaSCSI のブリッジ機能を利用したネットワーク・ファイル共有デバイスドライバ
RASETHER は SCSI バスを通して Raspberry PI とのネットワーク通信を行い、RASDRV は Raspberry PI のファイルシステムを X68k で論理ドライブとしてマウントして、他にネットワークと繋がったマシンと自在にファイルが交換できる

RaSCSI の配布ファイルの中にドライバが含まれています

・RASCPY
sava さん(Twitter：@lpproj)
https://github.com/lpproj/rascsi.uty/releases

X68k 用ファイル共有ドライバである RASDRV の PC-98 版

とはいっても、68 版のように論理ドライブが割り当てられるとかではなく、どっちかというと FTP クライアントのような存在だが、RaSCSI のブリッジ機能を用いて Raspberry PI とのファイルがやり取りできるという特徴は一緒。これがあるだけで Windows マシンから 98 へのファイル交換が、フロッピーやシリアル接続経由という面倒な従来方法から一気に便利になる

使用するのに ASPI ドライバが必要(フリーソフトで対応可能)

他に FreeDOS の PC-98 対応版も開発しておられるので、いずれ触ってレポートしたい

・RaSCSI／FDX68 対応イメージ切替フロントエンド(名前不明)
z α 2 さん(Twitter：@z_alpha2)
https://onedrive.live.com/?authkey=%21AIEpzizkVQ_rsK0&id=8E5428FE5454612E%215201&cid=8E5428FE5454612E

RaSCSI／FDX68 のイメージを web ブラウザ上から切り替える php をこっそり？公開されています(進化中)

動作イメージはこんな感じ→https://twitter.com/z_alpha2/status/1058430110531903488
Raspberry PI に web サービス周りを導入するのがちょっと難しいですが、これをクリアできれば今まで SSH でログインして、コマンドラインからイメージデータを切り替えるという作業から開放されます

この辺の便利ツール導入方法はいずれ「RaSCSI 応用編」として別途ドキュメントでまとめたいと思います

お・く・づ・け

「最低限これだけ設定すれば、とりあえず動かすことが出来る！はじめての RaSCSI 導入マニュアル」
(2018 年 12 月 25 日版)

著者：シエル(サークル Project M.P.S)　発行所：Project M.P.S サポートサイト(http://projectmps.net)
著書に関するフィードバックなどの連絡先 E-Mail：project_mps@outlook.jp Twitter：@MPS_support)

本書は RaSCSI テクノロジの発展の為にじゃんじゃん転載してください、著者に対する転載のお伺いなど不要です。ただ、ちょっと感想とか送ってくれると喜びます。キツーイお仕置きご意見はオブラードに包んで教えてください