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

2019/03/11 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(ベアメタル版の登場でかなり導入の敷居は下がりましたが)

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**
- **あいぼむ版**
- **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

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

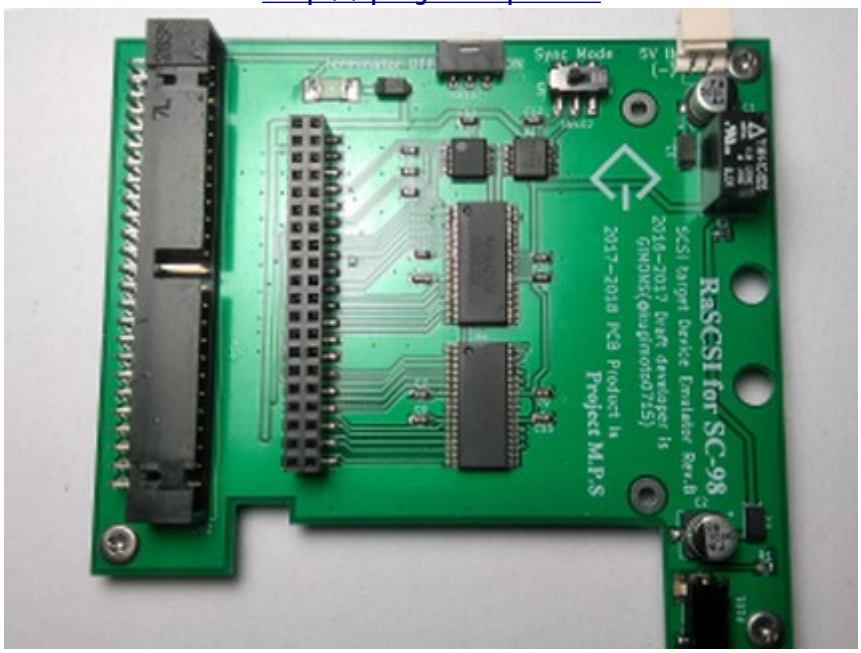
Version 2

「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

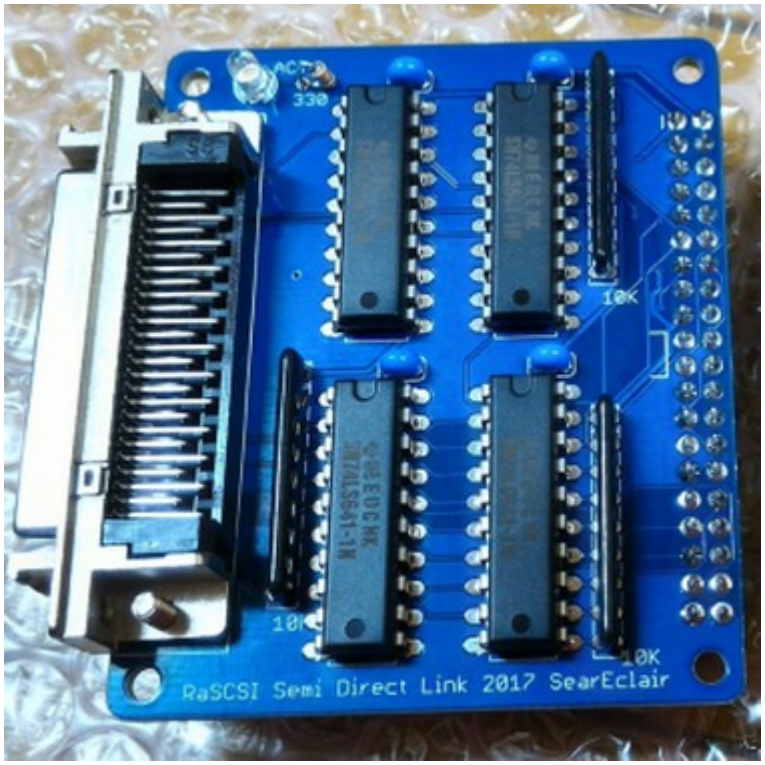
RaSCSI for SC-98 は後発ですが、あいぼむ版と同じ制御チップを使っているため、あいぼむ版互換になっています

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

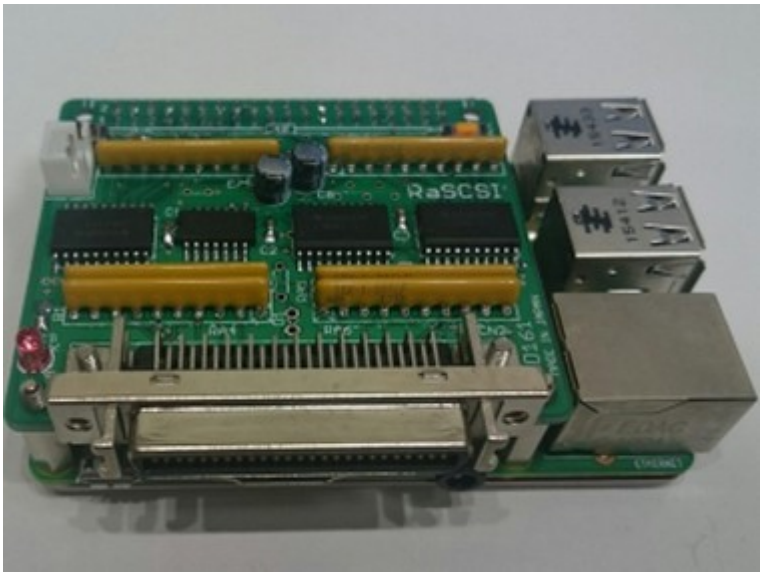


あいぼむ版
「RaSCSI Adapter Lite」(Rev.297)

http://products.aibom.net/ral297/

mobile 版
「ND161」

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



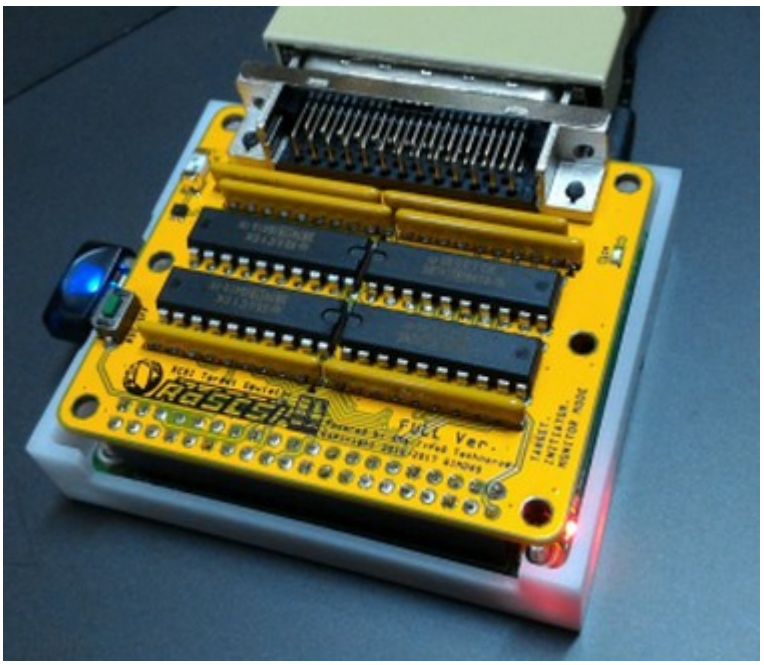
これらは全て「スタンダード版」のバイナリを使用します

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

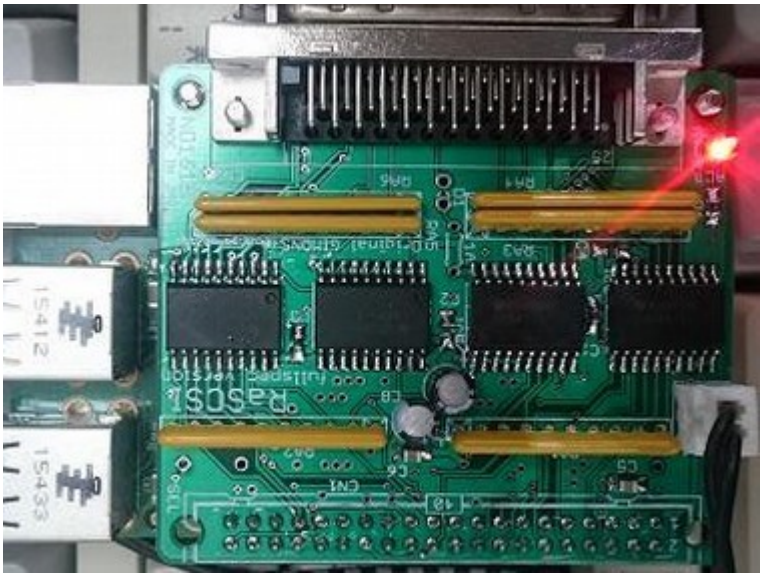
えくしみえむ版

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



mobile 版
「ND161B」

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

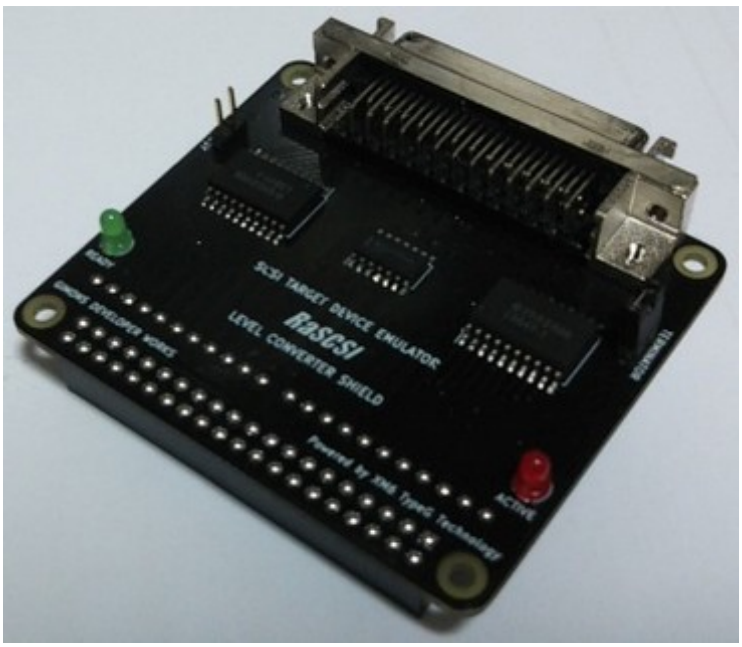


Project M.P.S 版「RaSCSI ZERO」

http://projectmps.net/

GIMONS DEVELOPER WORKS 版(公式版)
「レベルコンバータシールド」

http://retropc.net/gimons/

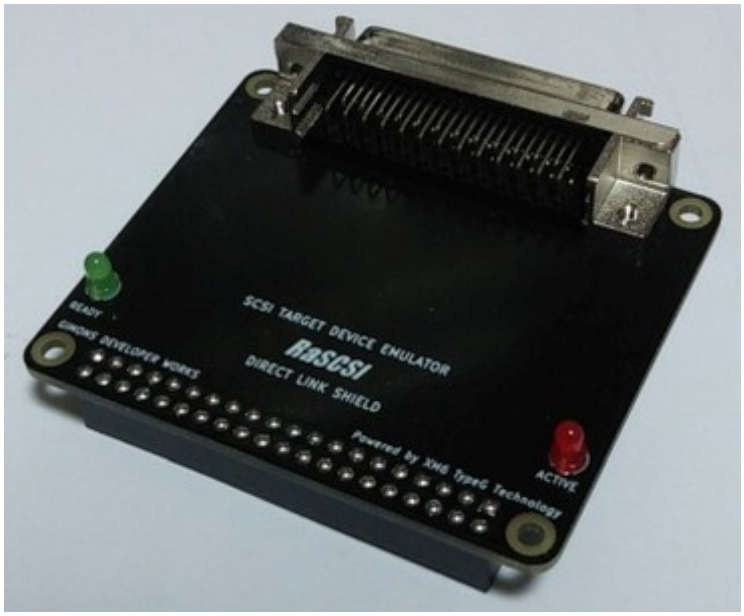


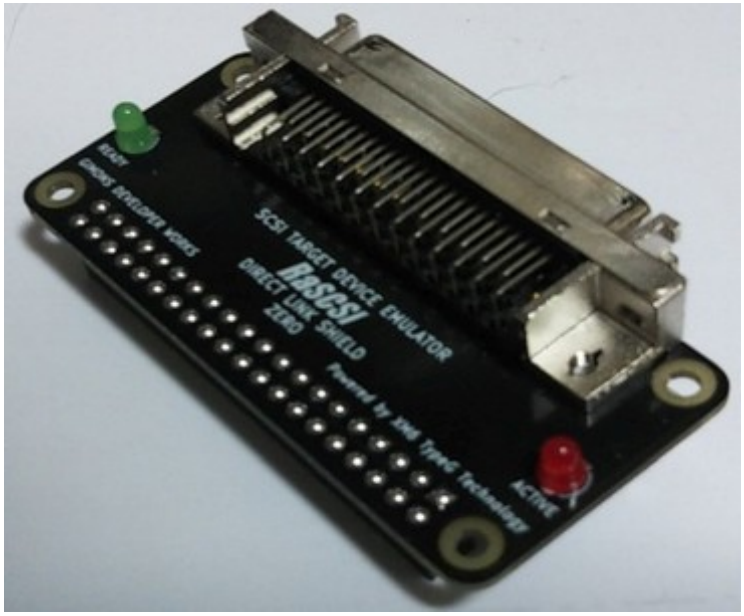
これらは全て「フルスペック版」のバイナリを使用します

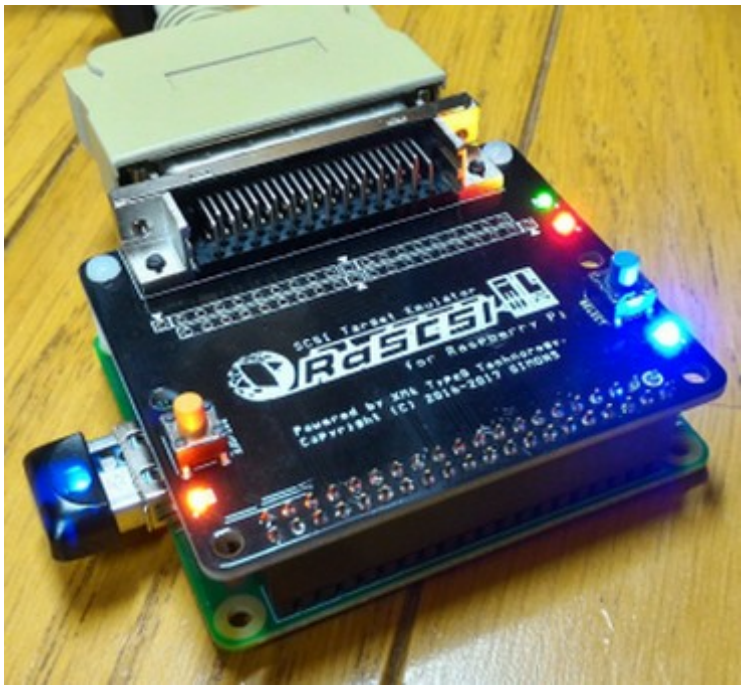
えくしみえむ版はスタンダード版と見分けがつきにくいですが、スイッチの数が減っていたり、LED 周りが変わっていたり「FULL Ver」のシルクが追加されていたりしますので何となく分かると思います。こちらも設計データが公開されているため、色違いの派生が存在する可能性があります

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

フルスペック版は現在でも頒布されている為、わりと多く出回っています(えくしみえむ版は基板データが)







その他「直結系」

上：GIMONS DEVELOPER WORKS 版
中：えくしみえむ版　下：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

バイナリは「スタンダード版」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 RaSCSI の運用形態について説明をしていきます

～2章 RaSCSI を運用する形態を決める～

2018 年までは RaSCSI を使用するためには、Raspberry PI 用の OS をインストールし、最初に各種設定をした上でそこに RaSCSI プログラムをインストール～さらに設定するという形しかなく、Linux を触ったことがない初心者にとっては、使えるまでの敷居が非常に高いものでしたが、2019 年になり RaSCSI の「ベアメタル版」というものが登場しました

ベアメタルとは早い話 Raspberry PI を「マイコン」として扱い、直接ハードウェアをコントロールして機能を実装する方法です

OS を介さないので起動が速く、ファイルシステムもシンプル、マルチタスク OS に必要なシャットダウン処理も必要ないので、使い終わったらいきなり電源を切ることも可能です(マルチタスク OS はいきなり電源を切ると色々データが壊れたりする)

代わりに、OS ありきで実現されている機能、例えばネットワークや USB まわり、Raspberry PI ならカメラやモニタなどの機能も自前で実装しなければならず、仕様が公開されていない部分は実装できないので使用することができません

ベアメタル版の利点・欠点は公式ページで言及されています→http://retropc.net/gimons/rascsi_bm/index.html

OS(Raspbian)版の利点
・ネットワーク経由による外部からのコントロール(SSH)
・ネットワーク経由による別マシン間でのファイル共有(samba)
・USB ポートが使える(それに伴う各種ドライバも自動的に組み込まれる)
・自前で空ディスクイメージが作れる(dd コマンド)

欠点
・OS ブートに時間がかかる
・マルチタスク故の急な電源断によるシステム破損の危険性
・システムを破損させた場合の復旧に手間がかかる(ヘタすると OS インストールから)
・起動しなくなったシステムからディスクイメージを救出するために Windows の場合ひと手間必要になる
・Raspberry PI Zero など非力なハードウェアの場合、タスクスケジューリングにリソースを取られ、
ほかのプログラムの動作に影響を与えることがある

ベアメタル版の利点(ほぼ上記ホームページの受け売り)
・起動が高速
・シャットダウンフリー
・簡単セットアップ(プログラムを SD カードにコピーするだけ)
・簡単イメージ転送(SD カードにコピーするだけ)
・簡単バックアップ(FAT32 ファイルシステム)
・RaSCSI プログラムの処理にハードウェアリソースをすべて使えるために
Raspberry PI Zero など非力なハードウェアでも安定動作しやすい

欠点
・ネットワークが使えない
・USB ポートも使えない
・ディスクイメージ作成は外部のツールに頼る
・ディスクイメージを切り替える時は、一度電源を落として SD カードを別のマシンに繋ぎ、
テキストエディタで書き換えた上で再び電源を入れる必要がある(固定運用向き)

どちらも一長一短なので、運用方法に応じて決定してください。従来のSCSIハードディスクとの置き換えならばベアメタル版を、柔軟にディスクイメージを変更しながら使用したい場合は OS 版をオススメします

次はベアメタル版・OS 版共通に必要な物や下準備の解説をしていきたいと思います

最初に必要な物(ベアメタル版・OS 版共通)

・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～(ベアメタル版・OS 版)
※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・アダプタ(ベアメタル版・OS 版)

・給電用 microUSB アダプタ 2.5A 以上(ベアメタル版・OS 版)
※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(OS 版)
※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(ベアメタル版・OS 版)
※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(OS 版)

～3章その１　RaSCSI の設定　ベアメタル編～

ベアメタル版の設定については、公式ページ→http://retropc.net/gimons/rascsi_bm/index.html にも触れられていますし、そう難しい設定もありませんが、軽く触れておきます

・microSD カードの初期化

新品のカードの場合には必要ありませんが、別の用途に使われていたカードを転用する場合は初期化作業が必要になります。これはベアメタル版・OS 版問わず必要な作業です

SD アソシエーションの「SD メモリカードフォーマッタ」をダウンロード→https://www.sdcard.org/jp/index.html
上部の「ダウンロード」→左側の「SD カードフォーマッター〇〇用(Windows or Mac)」を現在の使用環境に応じてクリックする

英文がズラズラズラ～と出てきますが、SD カード規格のライセンスや規約などの確認事項です。今回の作業には必要ない項目なので、サラッと流して最下段の「同意します」ボタンをクリックしてください

SDCardFormatterv5_WinJP.zip(Windows 版の場合)というファイルがダウンロードされます

展開すると SDCardFormatterv5_WinJP というフォルダが自動的に作られ、そのフォルダの中に
SD Card Formatter 5.0.1 Setup JP.exe という実行ファイルがありますので、インストールを行ってください

デスクトップ(またはタイル)に左のようなアイコンが登場しますので

先に PC に SD カードを接続してから

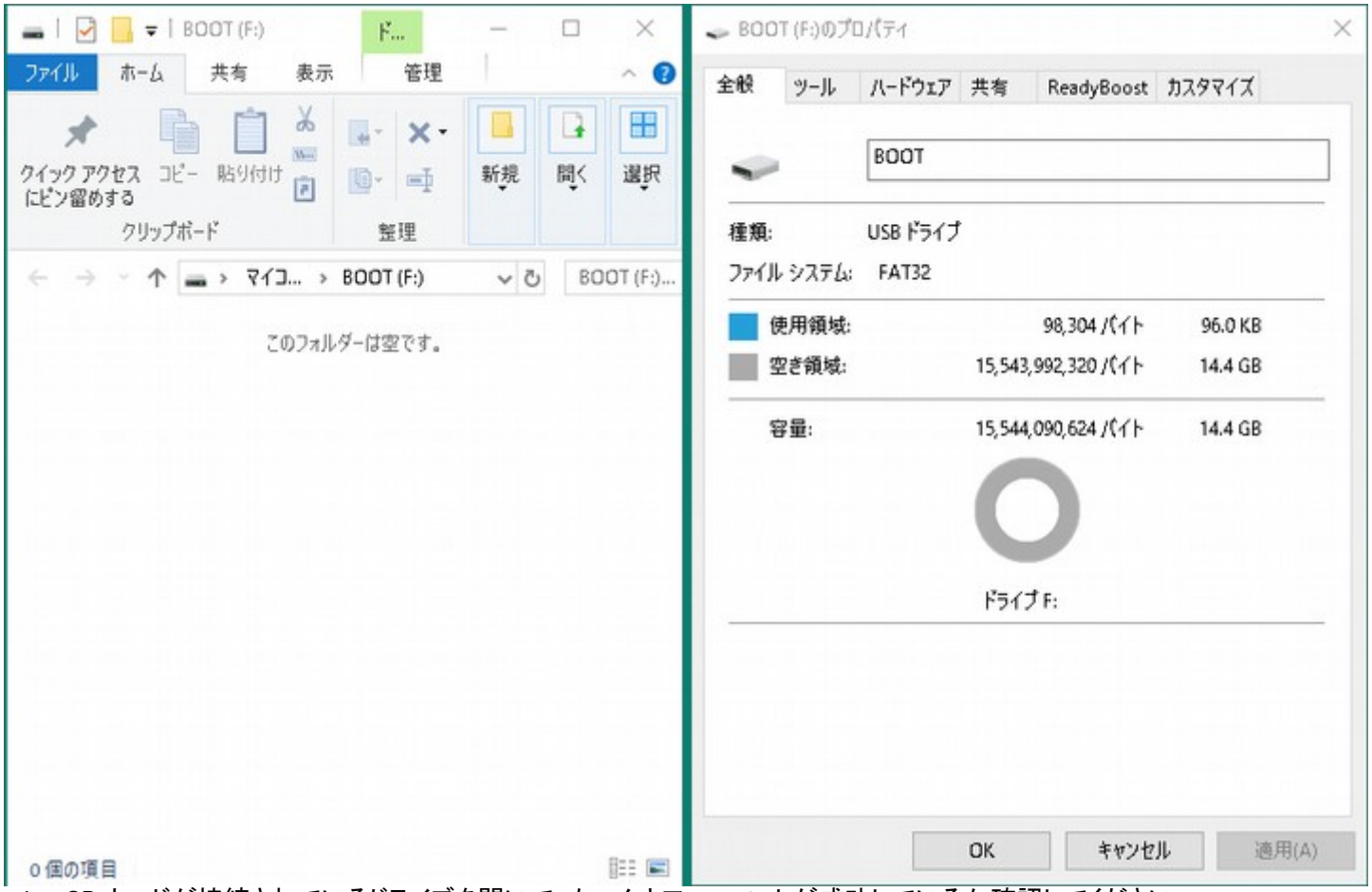
アイコンをクリックしてプログラムを実行してください
(まぁ実行後にカードを接続しても認識しますけどね)

このようなウィンドウが開きますので、フォーマットボタンを押してください
「フォーマットオプション」は基本的にクイックフォーマット(高速)で OK です
まれにクイックでフォーマットに失敗する場合がありますが、そういう場合は
上書きフォーマットで行ってください(クイックよりやや時間がかかります)

ボリュームラベルは好きな名前を付けてください

なお、当然ながらフォーマットすると SD カードのデータは
すべて消えますので必要なデータがある場合は

**必ずバックアップを取ってから
フォーマットしてください**



microSD カードが接続されているドライブを開いて、ちゃんとフォーマットが成功しているか確認してください

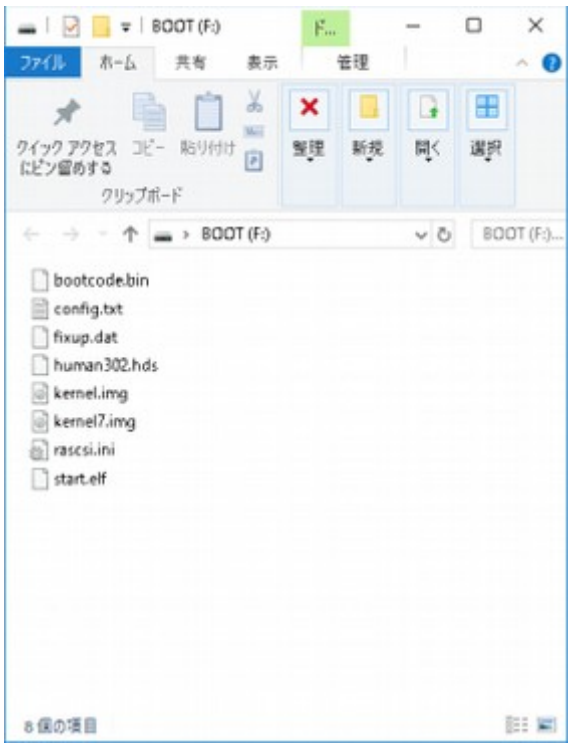
・RaSCSI(ベアメタル)のダウンロードと展開

続いて、RaSCSI(ベアメタル)の公式ホームページから各 RaSCSI アダプタ用のファイルをダウンロードします

RaSCSI(ベアメタル版)公式ページ→http://retropc.net/gimons/rascsi_bm/index.html

スタンダード版:RASCSI_BM_STANDARD.zip
フルスペック版:RASCSI_BM_FULLSPEC.zip
あいぼむ版:RASCSI_BM_AIBOM.zip
GAMERnium.com版:RASCSI_BM_GAMERNIUM.zip

「自分の持っている RaSCSI アダプタが何版のプログラムを使うか分からない」という方は 1 章をご覧ください



ダウンロードした圧縮ファイルを展開し、SD カードが接続されたドライブに展開します(フォルダごとではなく、中身を展開してください)

中身のファイルは全タイプファイル名が同じなので、複数のタイプの RaSCSI アダプタを持っている場合は SD カードが混ざると分からなくなるので、 何か識別できるような目印を用意しておくといいかもしれません

通常タイプ Raspberry PI と Raspberry PI Zero などのハード的差異はプログラムの方で吸収してくれるので、ユーザー側で意識する必要はありません

・RaSCSI のディスクイメージを作る

OS 版では「dd コマンド」で各種ディスクイメージを作ることができますが、ベアメタル版にはその機能がないので、外部のツールを使いディスクイメージを作る必要があります

方法としては
・XM6 TypeG を使う(RaSCSI の本来の想定される使い方)
・T98-Next を使う(PC-98 エミュレータその 1)
・Anex86 を使う(PC-98 エミュレータその 2)
・別の Raspberry PI(OS 版)を使うか、Linux が動いているマシンで作って転送

全方法をここで書き出すと大変長くなるので、以下のページをご覧ください

Project M.P.S「RaSCSI で使えるディスクイメージを作る」
http://projectmps.net/rascsi_img.htm

なお、この方法はベアメタル版のみならず、OS 版にも応用できます

・作ったディスクイメージをマウントする

先ほど SD カードに転送したファイル群の中に[rascsi.ini]というファイルがあります、これが設定ファイルです

中にサンプルがあるので、記述方法に迷うことはないとは思いますが一応書いておきます

```
ID0 human302.hds
ID6 bridge
```

使えるディスクイメージ形式は OS 版 RaSCSI と同一です
HDF→標準 SASI 形式
HDS→標準 SCSI 形式
HDN→NEC PC-9801-55 形式
以下略、あとはこのドキュメントの末尾をご覧いただくか、RaSCSI のマニュアルをご覧ください

もちろん CD-ROM の ISO 形式や、MO イメージの MOS 形式も扱えますが、MO からブートできる機種(PC-98)などは他の起動できるデバイスを差し置いて最優先で起動してしまうので、リアルタイムにディスクを抜くことができないベアメタル RaSCSI では MO 形式は色々な意味で使い物になりません

・SD カードを RaspberryPI＋RaSCSI アダプタに接続する

パソコンの電源を入れて即使用可能になります、RaSCSI が動いているという意識をする間すら与えません

・使い終わったら

パソコンの電源を即落として大丈夫です。ただ、普通のハードディスクと同様ディスクアクセス中に電源を落とすと、ディスクイメージデータを破損する可能性があります(普通そんなことはしないと思いますが)

・ディスクイメージのバックアップなど

SD カードは FAT32 フォーマットなので、普通に拡張子 HDS などのディスクイメージデータを別途 USB メモリやハードディスクにコピーすればいいだけです

～3章その2　Raspberry PI 用 OS「Raspbian」のインストールと初期の設定～

今度は 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

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

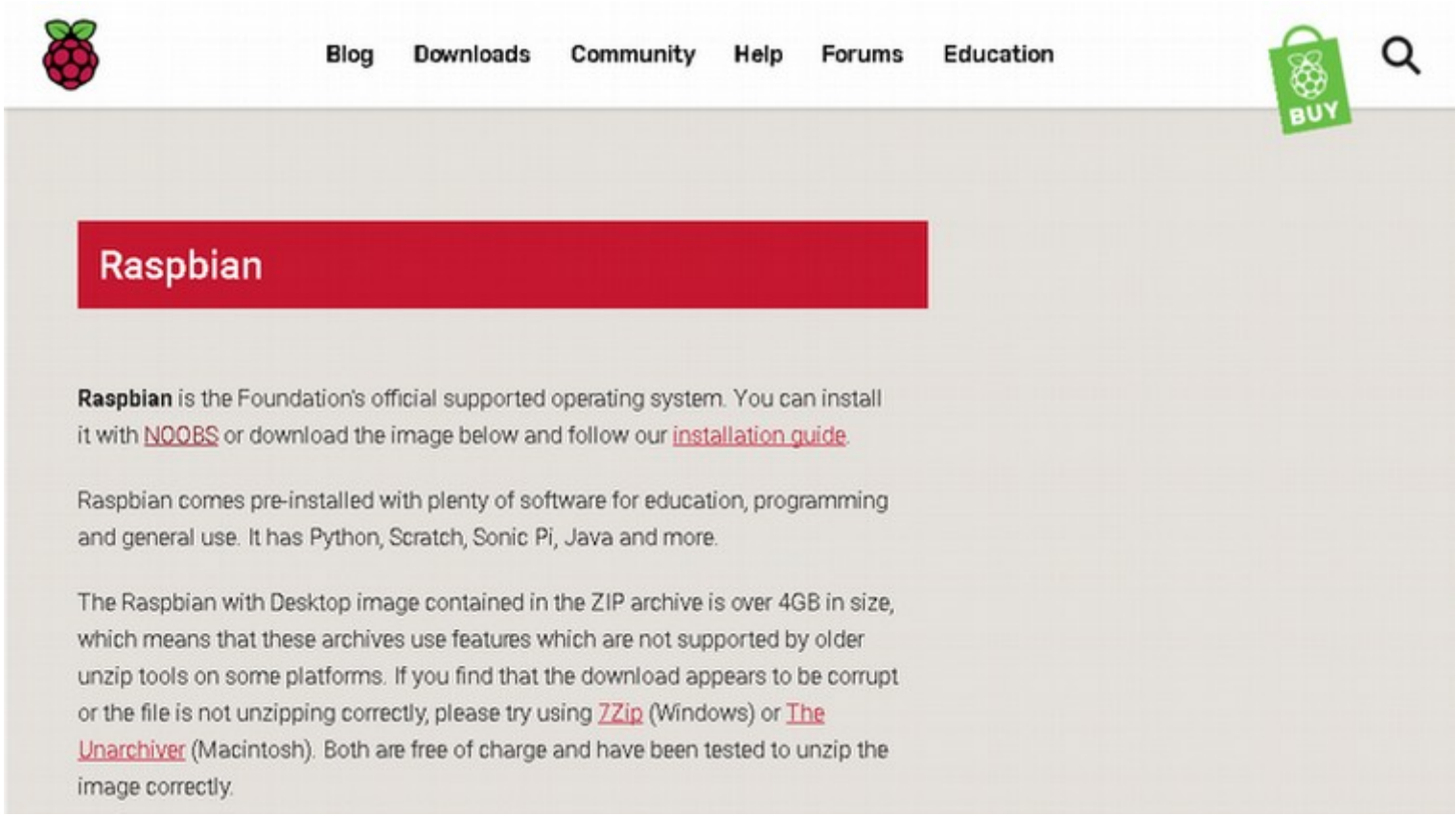
ただし、コンピューターに標準で OS などがインストールされていなかった当時と今の大きな違いは「OS が無料で手に入る」(もちろん通信費とか電気代は別ですよ？)ことと、インターネットの発達で「目的に対する情報が多すぎる」ことです

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

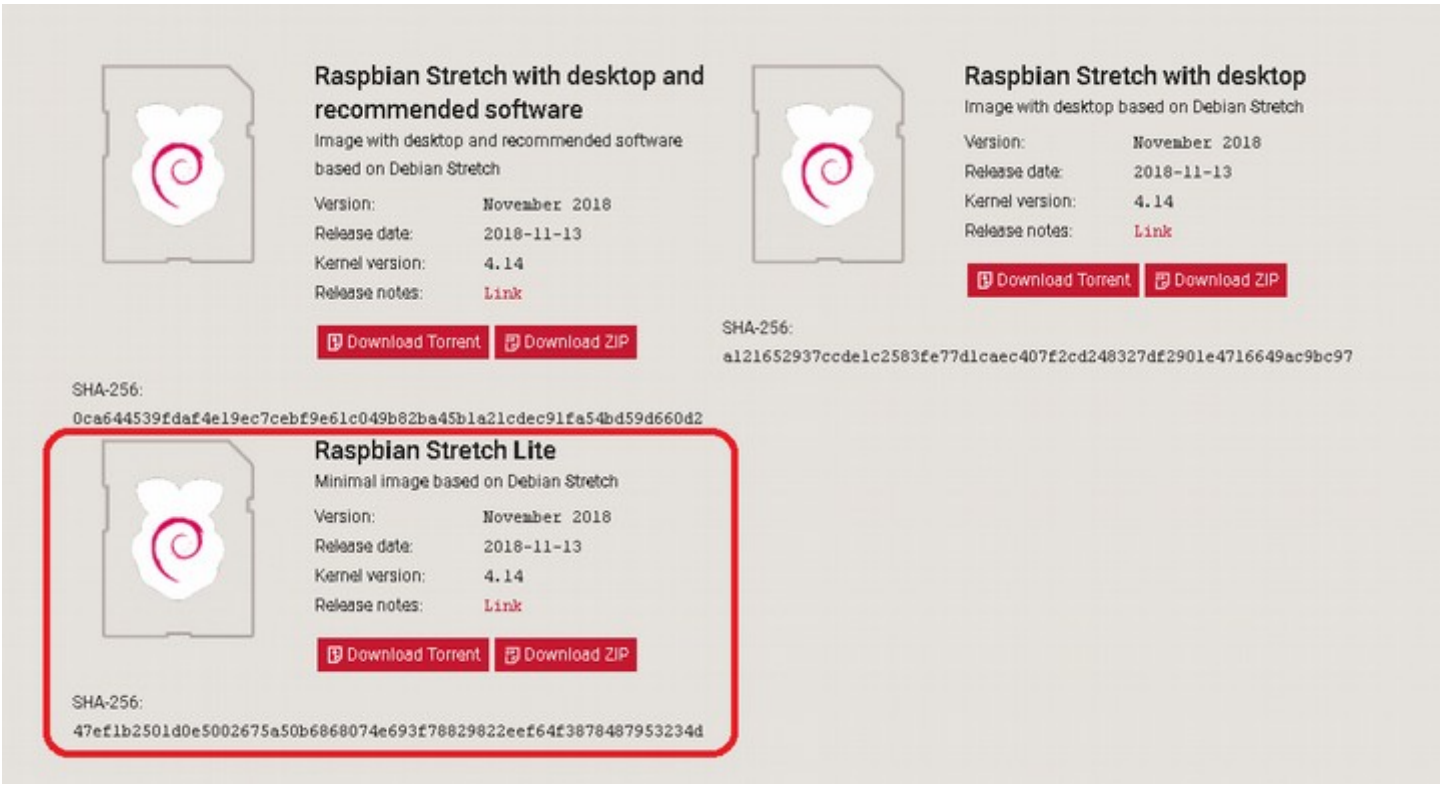
Raspbian のダウンロード

「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

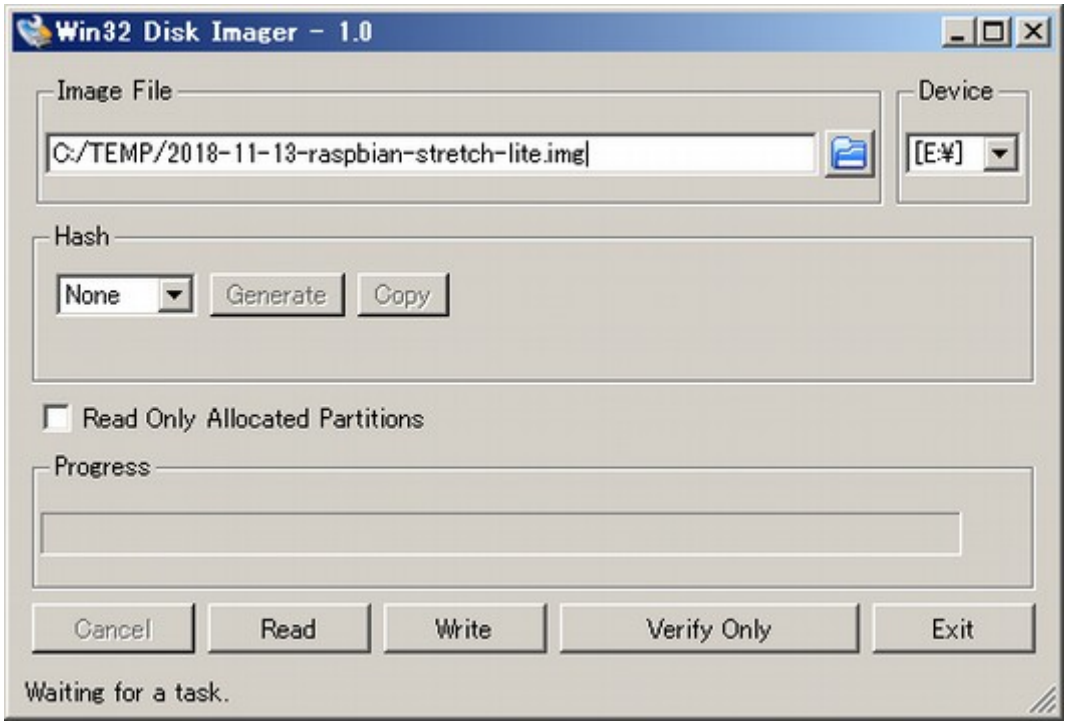
SourceForge.net はオープンソースプロジェクトの総本山みたいな所なので、そのテには厳しいですが、こういう造りの Web ページには一定の警戒感が必要です

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

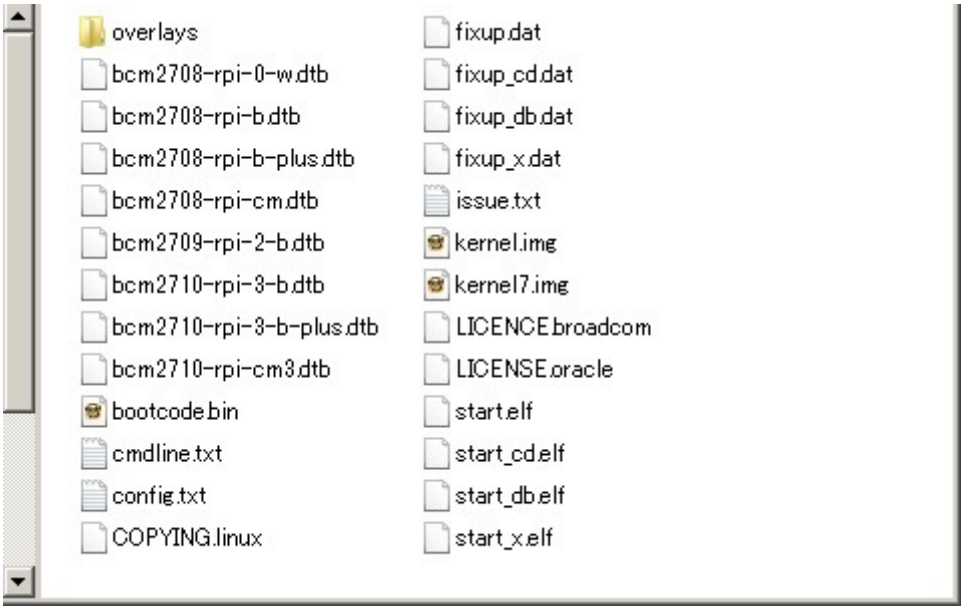


「**Image File**」が先ほどダウンロードした Raspbian の圧縮ファイルを展開したイメージファイルデータ

「**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

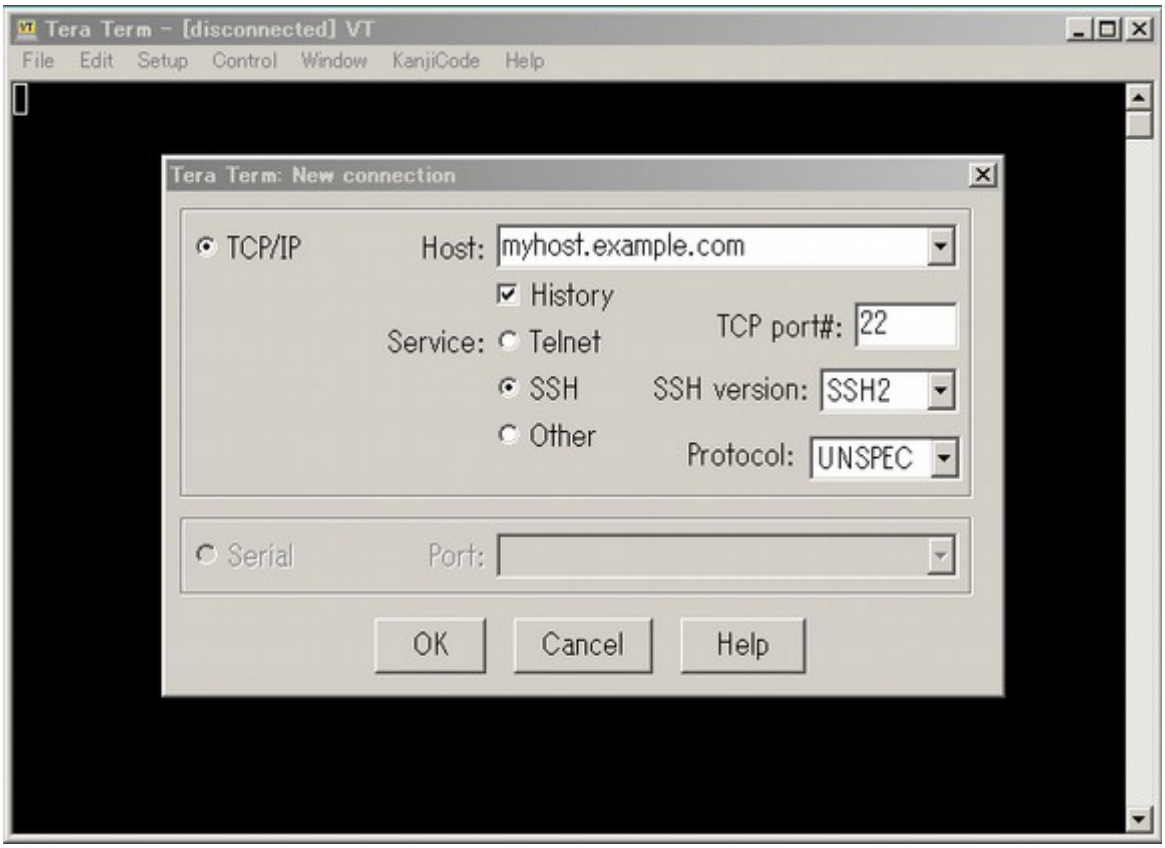
Tera Term
https://ja.osdn.net/projects/ttssh2/

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



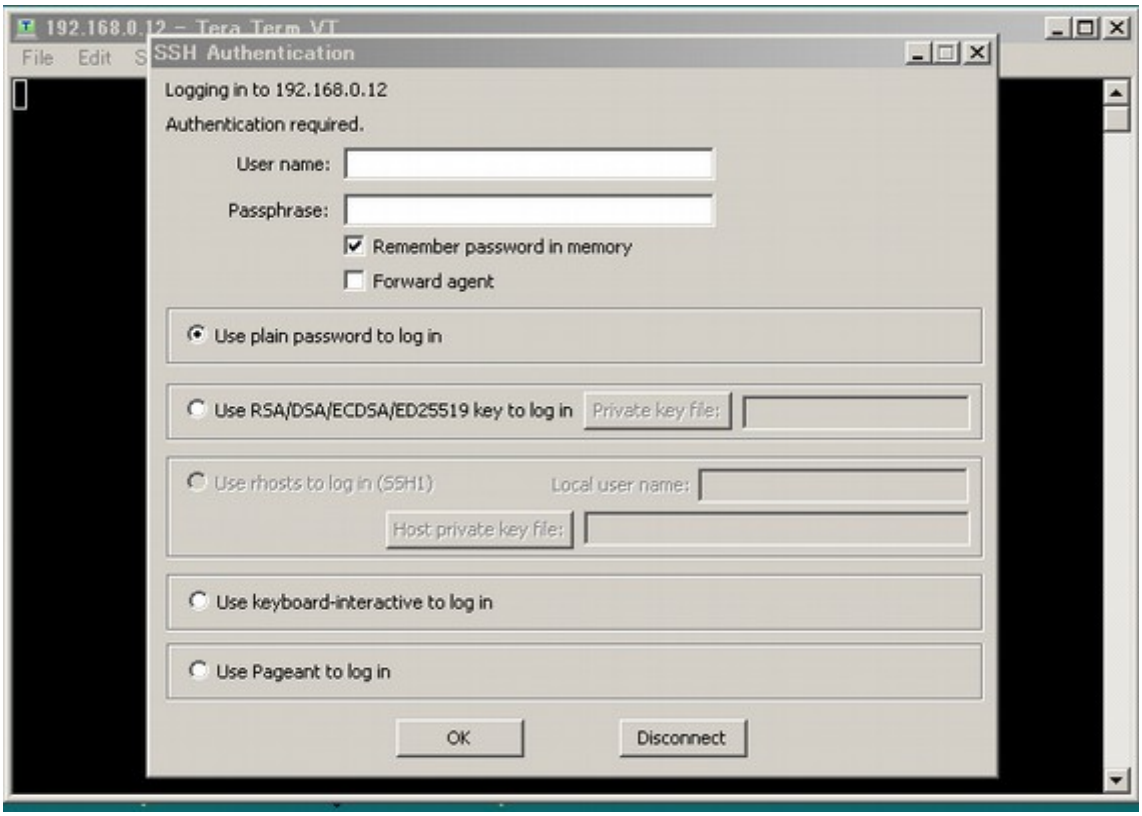
続いて Raspberry PI に割り当てられている IP アドレスを探します。本来なら Raspberry PI をモニタに繋いで、接続されたキーボードから「ifconfig」コマンドで自 IP アドレスを知るのですが、最初からモニタレスで SSH 接続を前提にする場合はその方法が使えませんので、**Windows のコマンドプロンプトから ping コマンドで総当りをします**

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)

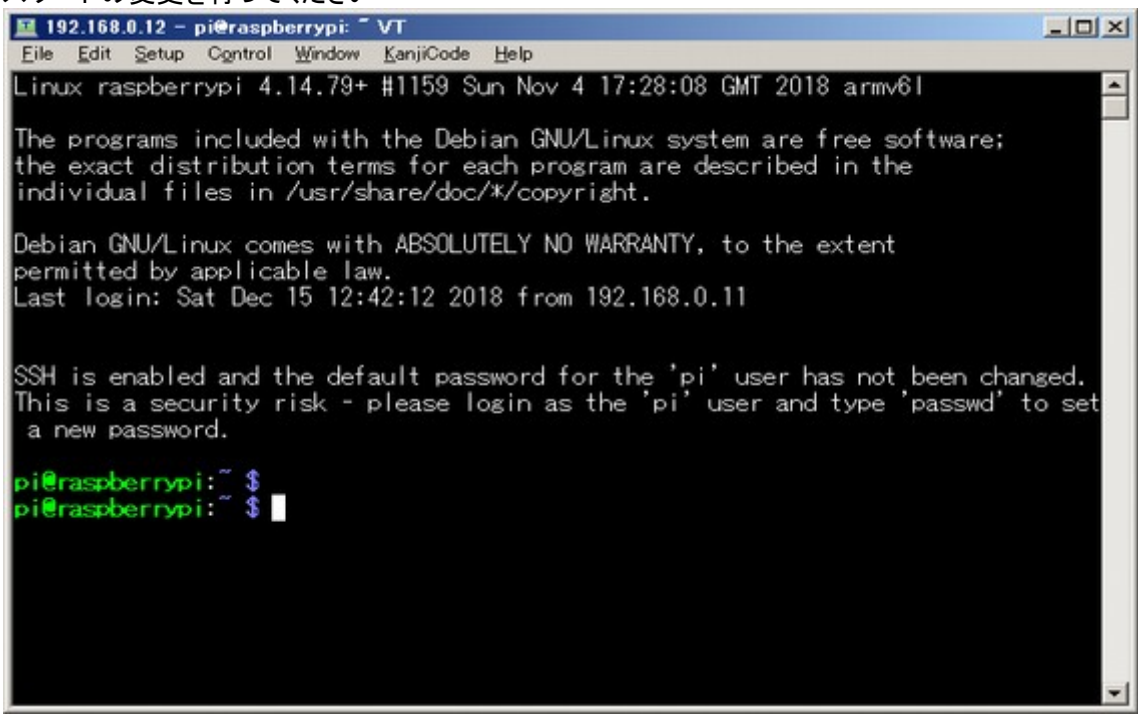
基本的に後から起動したネットワークデバイスが、すでに起動しているネットワークデバイスの IP アドレスの前に来ることはまずないので、そのマシンの IP アドレスを起点として、**ping コマンド**で末端の数字を 1 づつ増やしてエコーが返ってくるアドレスを探します

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

今まではこれから紹介するコマンドラインをひとつひとつ打ち込んでいく設定方法で Raspberry PI(Raspbian)の下準備～RaSCSI プログラムのインストールと設定という流れで進めていきましたが「RaSCSI for php」の作者である Zα2 さん(Twitter:@z_alpha2)が Raspbian のアップデート関連～RaSCSI プログラムのダウンロード＆展開～自動起動設定まで一気に設定できる半自動スクリプトを開発されました

半自動ゆえ、処理の流れがつかみにくいことや、RaSCSI プログラムやディスクイメージの置き場所がスクリプトを書き換えない限り規定設定に固定されるなどありますが、とにかく難しいことはできる限りやりたくない、でも RaSCSI は使いたい！という方にはオススメできる方法です

SSH ウィンドウ画面上で
~ $ **wget https://raw.githubusercontent.com/ztto/rascsi-php/master/raspberrypi_setup.sh**
半自動実行スクリプトを github からダウンロード終わったら
~ $ **chmod 755 raspberrypi_setup.sh**
スクリプトファイルに実行権限を付ける
~ $ **sudo ./raspberrypi_setup.sh**
管理者権限でスクリプトを実行

実行すると

```
対象となるボードを選んでください.
1:RaSCSI
2:FDX68
q:終了
```

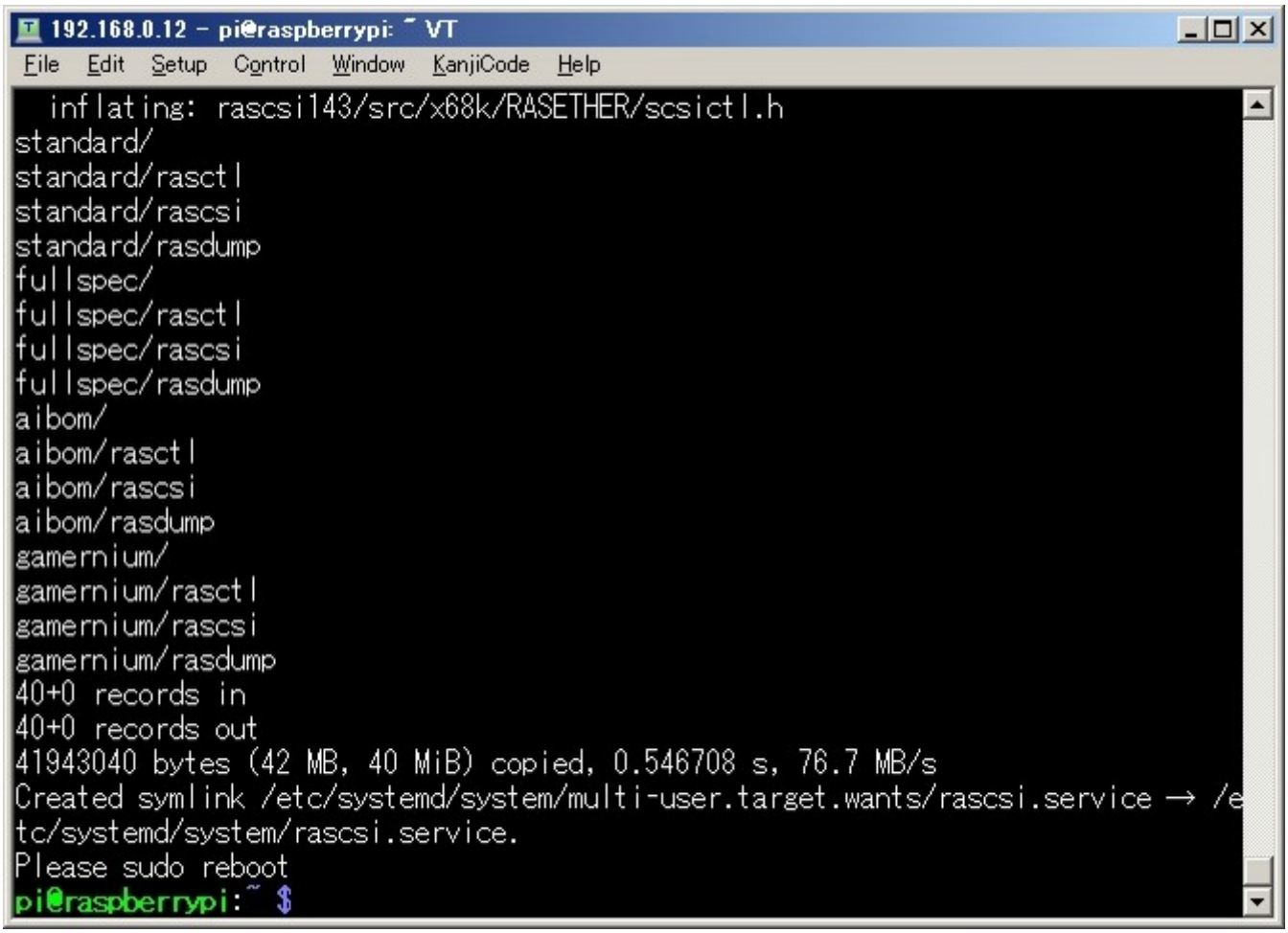
という選択肢が出るので 1:RaSCSI を選択し Enter キーを押します

続いて RaSCSI アダプタの種類を選択します

```
対象となるボードを選んでください.
1:standard 版
2:fullspec 版
3:aibom 版
4:gamernium 版
```

ここは「1章 RaSCSI アダプタの種類」に従って、現在お手持ちのアダプタタイプを選択してください

これら選択が終わると、Raspbian のアップデート作業や samba モジュールのインストール作業などが始まります。
修正モジュールの量によっては結構な時間がかかりますので、進行画面を横目に見つつ、何か他のことをやっていましょう



すべての作業が終わり、このような画面になったら Raspberry PI を再起動します

~ $ 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)

これで RaSCSI プログラムのインストールから自動実行までが設定されました。再び SSH で Raspberry PI に接続し

~ $ rasctl -l

と入力して、以下のような表示が出れば RaSCSI プログラムが自動起動されています

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

表示が確認出来たら Raspberry PI をシャットダウンし、電源を完全に切った上で RaSCSI アダプタをコンピューターの SCSI ボードに接続して、改めて Raspberry PI の電源を入れ実機で動作確認を行ってください

~ $ sudo shutdown -h now

おまけ：半自動スクリプト実行時の各種ファイルの設置場所など

RaSCSI プログラム→/usr/local/bin

各種ディスクイメージ→/home/pi/rasimg

・初期状態のディスクイメージほか
　RASDRIVER.HDS(10MB)　→RaSCSI 1.43 に添付されているイメージ、X68k 用。中身は RASDRV や RASETHER など
　scsiimg0.hds(40MB)　→半自動スクリプトで生成されたディスクイメージ、中身はブランクイメージ、フォーマット必要
　rasmount.sh　→　Raspberry PI の起動時にディスクイメージを自動マウントするスクリプト

　独自のディスクイメージを作成する場合は、上記のベアメタル版項目の外部ツールを使う方法か、下記の dd コマンドを使ったディスクイメージ作成を参照してください

RaSCSI 設定：ひとつひとつ動作を確認しながら設定していく方法

SSH ウィンドウ画面上で
~ $ **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ので OK)

Raspberry PI の根幹にかかわるシステムの設定を行います

~ $ **sudo raspi-config nonint do_expand_rootfs**
Raspbian のファイルシステムを SD カードの空き容量いっぱいまで広げる

~ $ **sudo raspi-config nonint do_change_timezone Asia/Tokyo**
Raspberry PI の時間設定を UTC(協定世界時)から JST(日本標準時)へ変更(UTC+9)

~ $ **sudo raspi-config nonint do_change_locale ja_JP.UTF-8**
ロケールを UTF-8.JP(日本語)に変更

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

現在のディレクトリの場所を確認します
~ $ **pwd**

もし、/home/pi 以外の場所にいる場合は、/home/pi に移動します
~ $ **cd /home/pi/**

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(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(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ていれば OK です

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 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

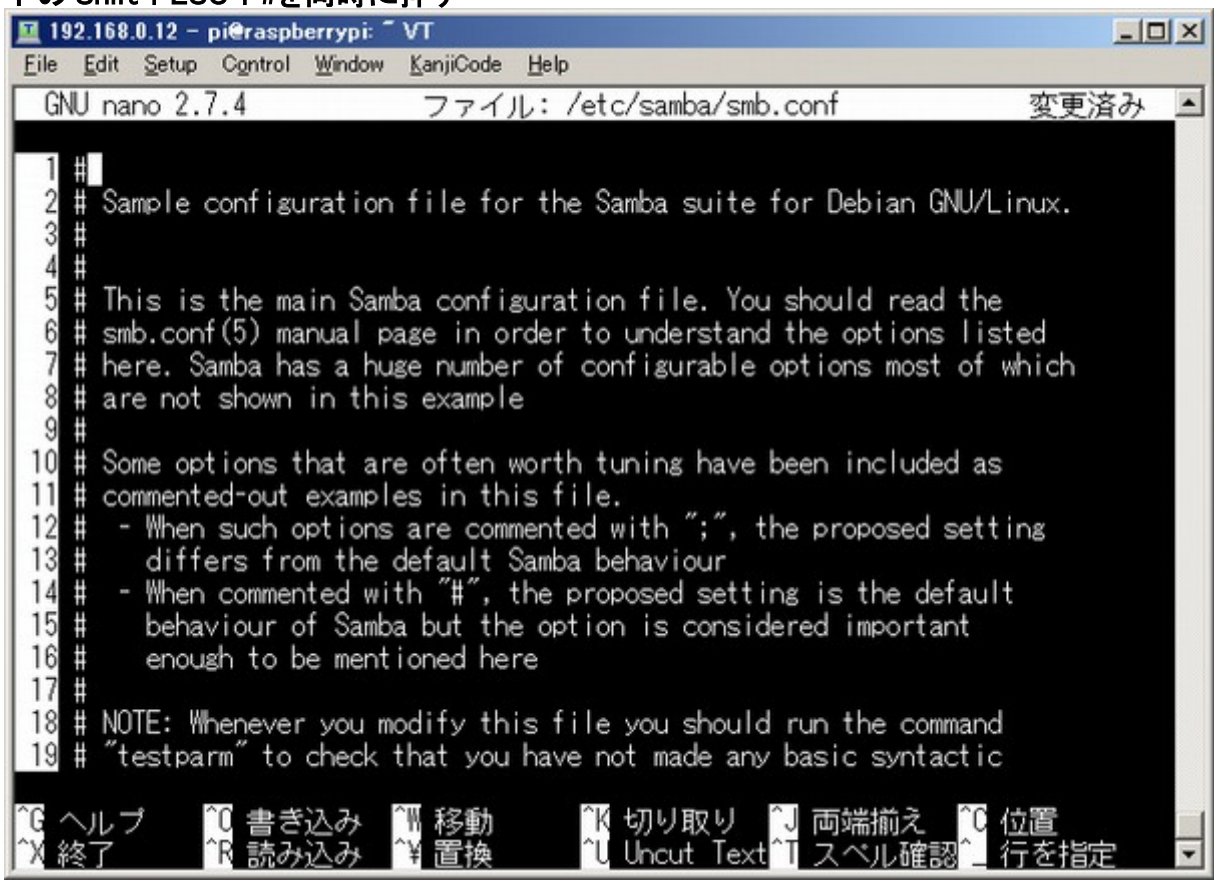
~ $ **sudo apt-get install -y samba**
インストール終了までちょっと時間がかかります、しばらくお待ちください

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
  force create mode = 0777
  force directory mode = 0777
```

[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

～4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

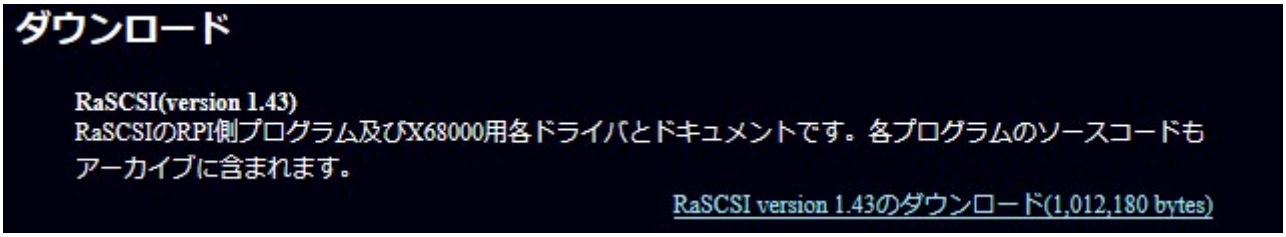
まず、Raspberry PI と RaSCSI アダプタを接続し、SCSI ケーブルをコンピューターに接続します。この際、Raspberry PI 側の電源供給手段は、コンピューター側のサービスコンセント等ではなく、必ず常時給電されているコンセントから取ってください

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://retropc.net/gimons/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

2019 年 3 月現在のバージョンは 1.43 なので、仮に **rascsi143** とします

```
~ $ mkdir rascsi143
~ $ cd rascsi143
```

pwd コマンドで、ちゃんと現在地が/home/pi/rascsi143 になっているか確認をします

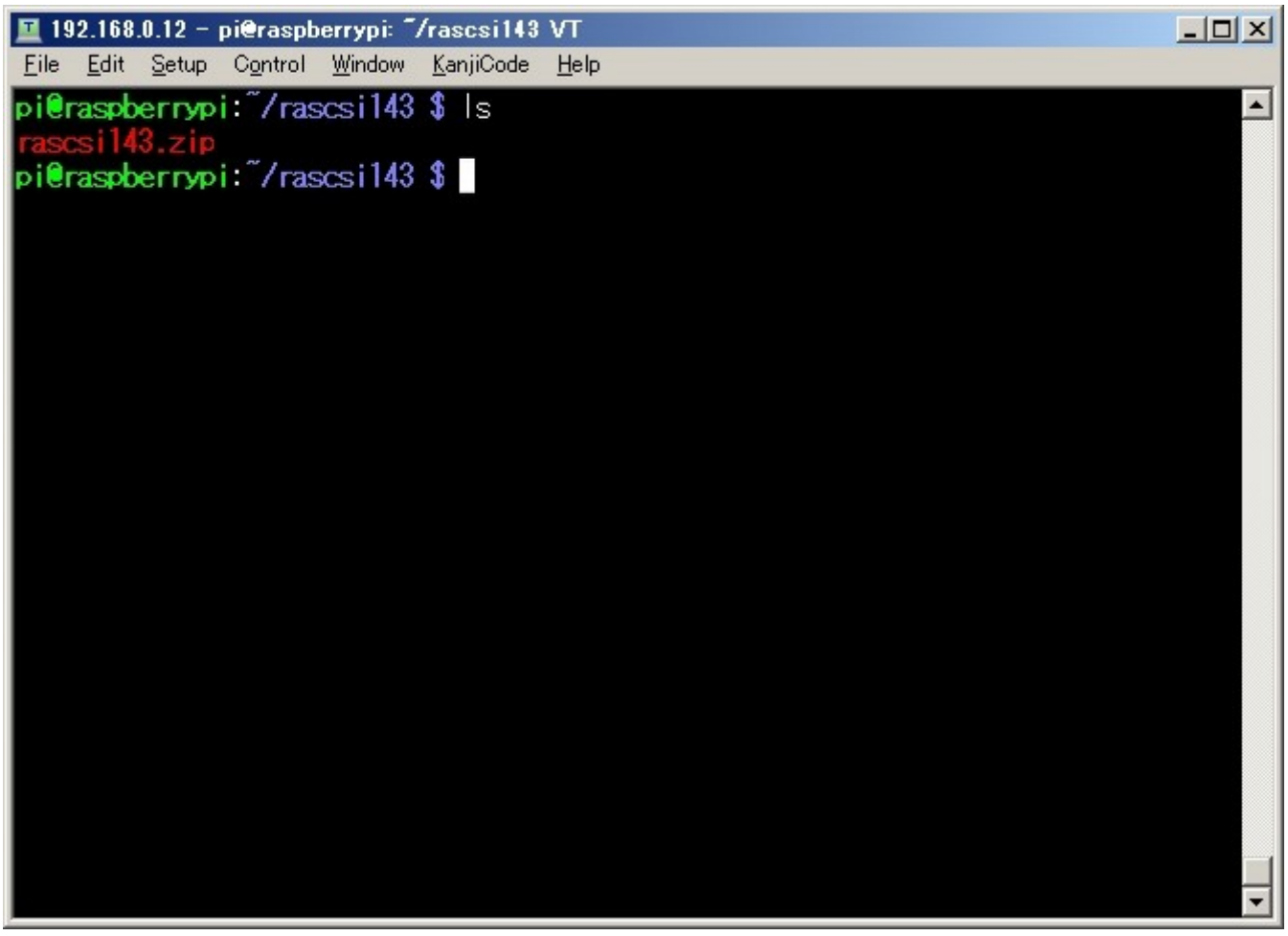
確認が出来たら、そこに RaSCSI の圧縮ファイルをダウンロードします。ダウンロードコマンドは **wget** コマンドです

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://retropc.net/gimons/rascsi/rascsi143.zip
```

※2019 年 3 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 rascsi143.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
rascsi143 rascsi143.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、rascsi143 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、このバージョン 1.43 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も同じという保証がないのと、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「**rascsi143**」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

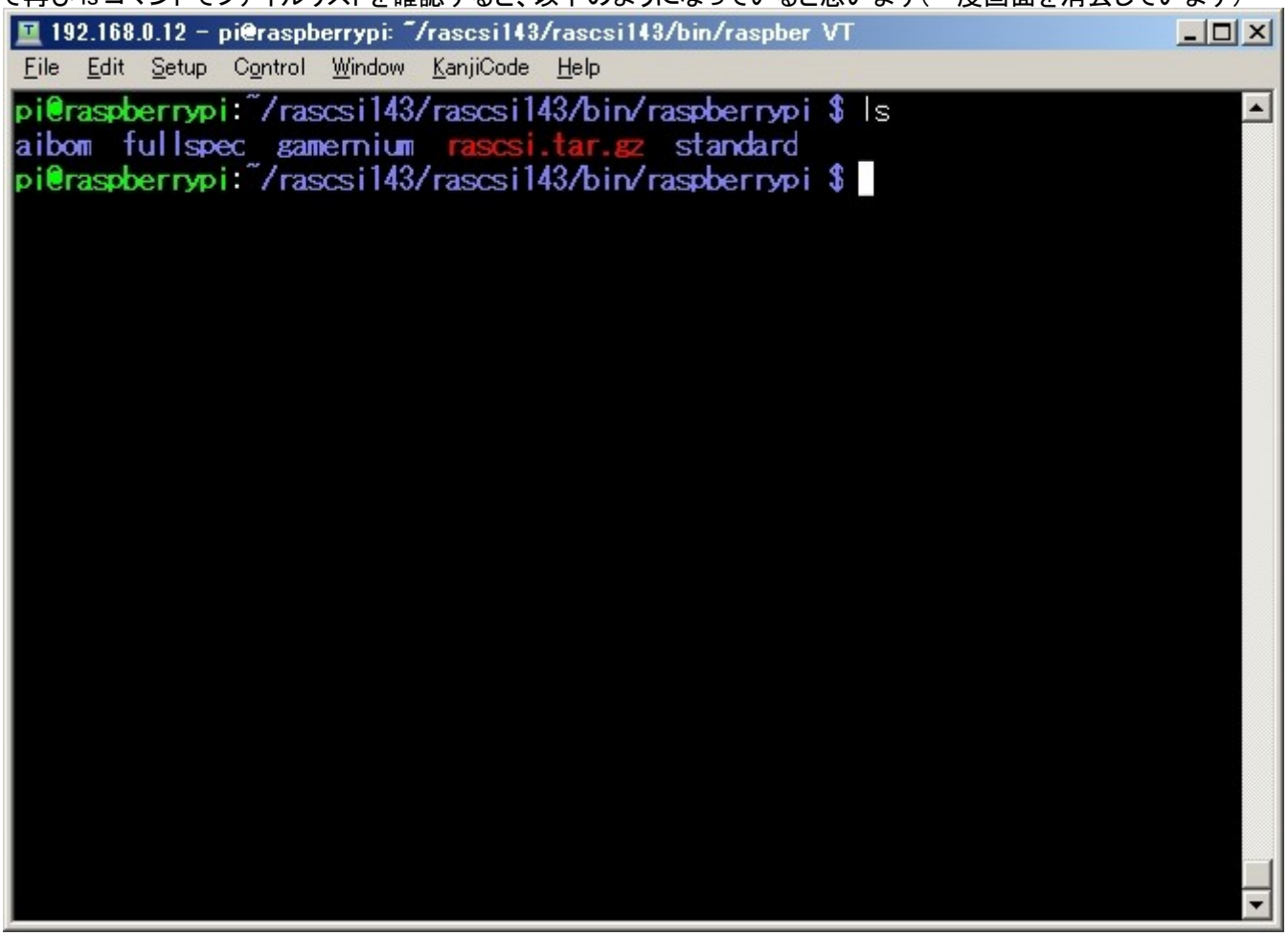
```
~ $ cd rascsi143/bin/raspberrypi
```

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

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 * /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

X68000 で使う場合は、68 で使える各種ドライバが納められたディスクイメージもコピーしておくと後々役に立ちます
~ $ cp ../../x68k/RASDRIVER.HDS /home/pi/rasimg/

確認が出来たらディスクイメージを置くディレクトリまで移動します(/home/pi/rasimg)

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/home/pi/rasimg ディレクトリに移動します

~ $ cd /home/pi/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 40MB のファイルが確認できればOKです
~ $ ls -l

(-rw-r--r-- 1 pi pi 10485760 1 月 23 00:00 RASDRIVER.HDS)
-rw-r--r-- 1 pi pi 41943040 1 月 23 00:00 scsiimg0.hds

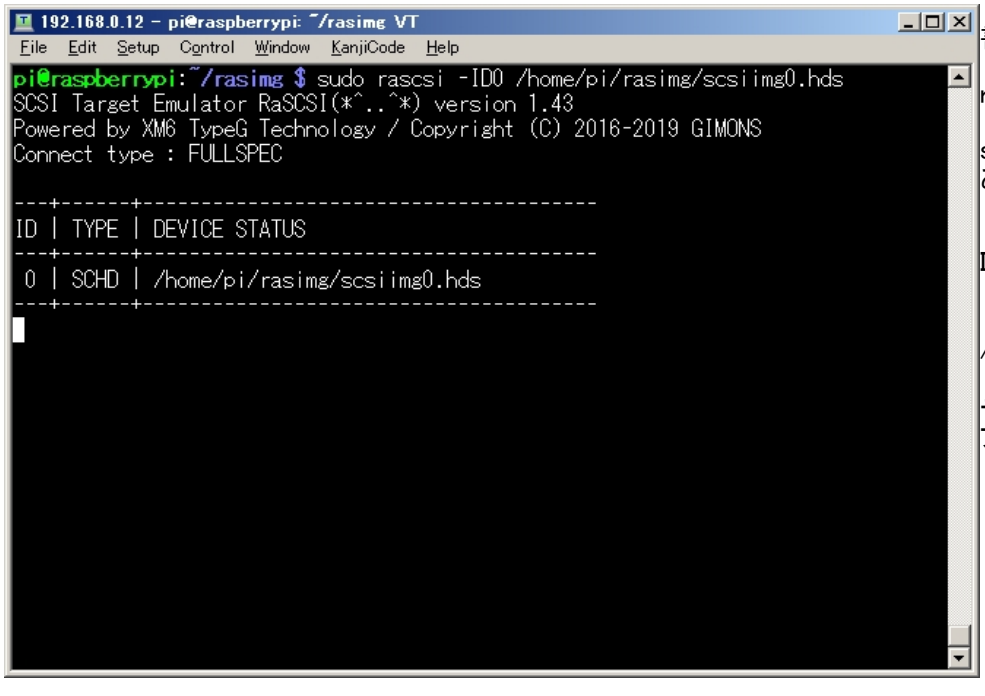
※括弧内は上記の X68000 用ディスクイメージをコピーされた状態

41943040 という数字を 1024 で 2 回除算してみると 40 という数字が出ると思います、これが 40MB の容量です

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

sudo(管理者権限)を付けて
あげないと起動しない

ID0 → 割り当てる SCSI-ID(今回は 0)

/home/pi/rasimg/scsiimg0.hds →

ディスクイメージまでのパスは
フルパスを指定してあげないとダメ

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 バージョン 1.43 ではカーネルドライバの組み込みが必要なくなりましたが、一部の SASI／SCSI ボードにおいてどうしても 1.43 では動作が不安定な場合があります。その場合は従来通りの Ver1.34＋カーネルドライバを適用してください
導入方法は「RaSCSI 導入マニュアル 2018/12/25 版」を参照してください

～5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 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

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

複数のイメージを記述する場合は
rascsi -ID0 ${hdir}scsiimg0.hds -ID1 ${hdir}scsiimg1.hds -ID2 ${hdir}cdrom.iso -ID6 bridge

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

いやもう、導入マニュアルの前版(2018/12/25 版)を発表してから、わずか 3 か月余で RaSCSI を取り巻く環境は大きく変わりました。2018 年までの RaSCSI は**「すごく便利だけど実際に使えるまでが大変なもの」**という扱いで、Raspbian(Linux)の前提知識がないと、そもそも RaspberryPI に OS のインストールすらままらない状態で、RaSCSI の導入支援サイトも Linux の知識がある人向けで、まったく触ったことがない人にとっては**「どこから手を付けていいか分からない」状態**

そんな状況に風穴を開けるべく、**「とりあえずこの通りにやれば RaSCSI を動かすことができる」**を目指して昨年末に執筆したのが、この「導入マニュアル」でした

「2018 年 1 年間、RaSCSI をとりまく環境は殆ど変わってないからこのマニュアルもしばらく改訂しなくて済むだろう」と余裕こいていたら、2019 年になって早々、設定上の一番の難所であったカーネルドライバ廃止版のバージョン 1.4 が発表され、ほぼ同時に OS のインストール作業などが不要で導入作業が相当簡便化されたベアメタル版もリリースされました

現在は細かいバグフィクスをされたバージョン 1.43 がリリースされています。それと同時に、今まで自分以外はほとんど静観状態だった RaSCSI 周りの議論が色んな意味で再沸騰し、頒布規約の変更や各 RaSCSI アダプタ製作者による動作確認状況の公表、公式版の RaSCSI シールドが頒布開始されるなど RaSCSI を取り巻く流れが大きく変わってきています

OS 版導入方法も、カーネルドライバ組み込みの作業が無くなったからといって、いちいち手作業でアップデートかけたり、samba 入れたり、自動実行周りの設定が結局いるからあまり手間は変わらなくない？と思っていたところに、Zα さんが開発された**「RaSCSI 半自動導入スクリプト」**が登場し、**「もうこの導入マニュアルいらなくない？」**と思っていましたが、各種 RaSCSI アダプタから使用バイナリを導き出す部分とか、Raspberry PI に OS をインストール～SSH や無線 LAN 設定とかの部分は自動実行スクリプトでは対応できない部分なので、何とか生き残っています(笑)

RaSCSI を取り巻く周辺プログラムも進化しています。公式版の X68000 で使える支援ドライバ RASDRV／RASETHER に続き、同じくZα さん開発の RaSCSI のディスクイメージを web ベースで切り替えられる「RaSCSI for PHP」や、RASDRV 相当を DOS マシンで実現する「RASCPY」に続き「RASDRV for ASPI(DOS 汎用)」「RASDRV for 55BIOS(PC-98 専用)などを sava さんが開発されています。このプログラムの存在は Windows マシンとのデータ交換の壁を劇的に低くする革新的なものです

この 3 か月間であまりにも RaSCSI 周りが変化したため、この導入マニュアルにも変化した部分をできる限り取り入れた結果、内容がかなりちぐはぐになってますが(汗)、それでもこの導入マニュアルで紹介されている所は自動実行による SCSI ディスクイメージの自動マウントまでで、その後の各種応用部分は限定的な取り扱いとなっています。内容的に足りない部分は、とりあえず自ホームページ(http://projectmps.net)で随時取り上げていきたいと思います

2019 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、ホスト名を変更後は一部の設定ファイルを書き換える必要があります、SSH 接続の際に名前解決にホスト名を使っている場合は、そちらのほうも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
```

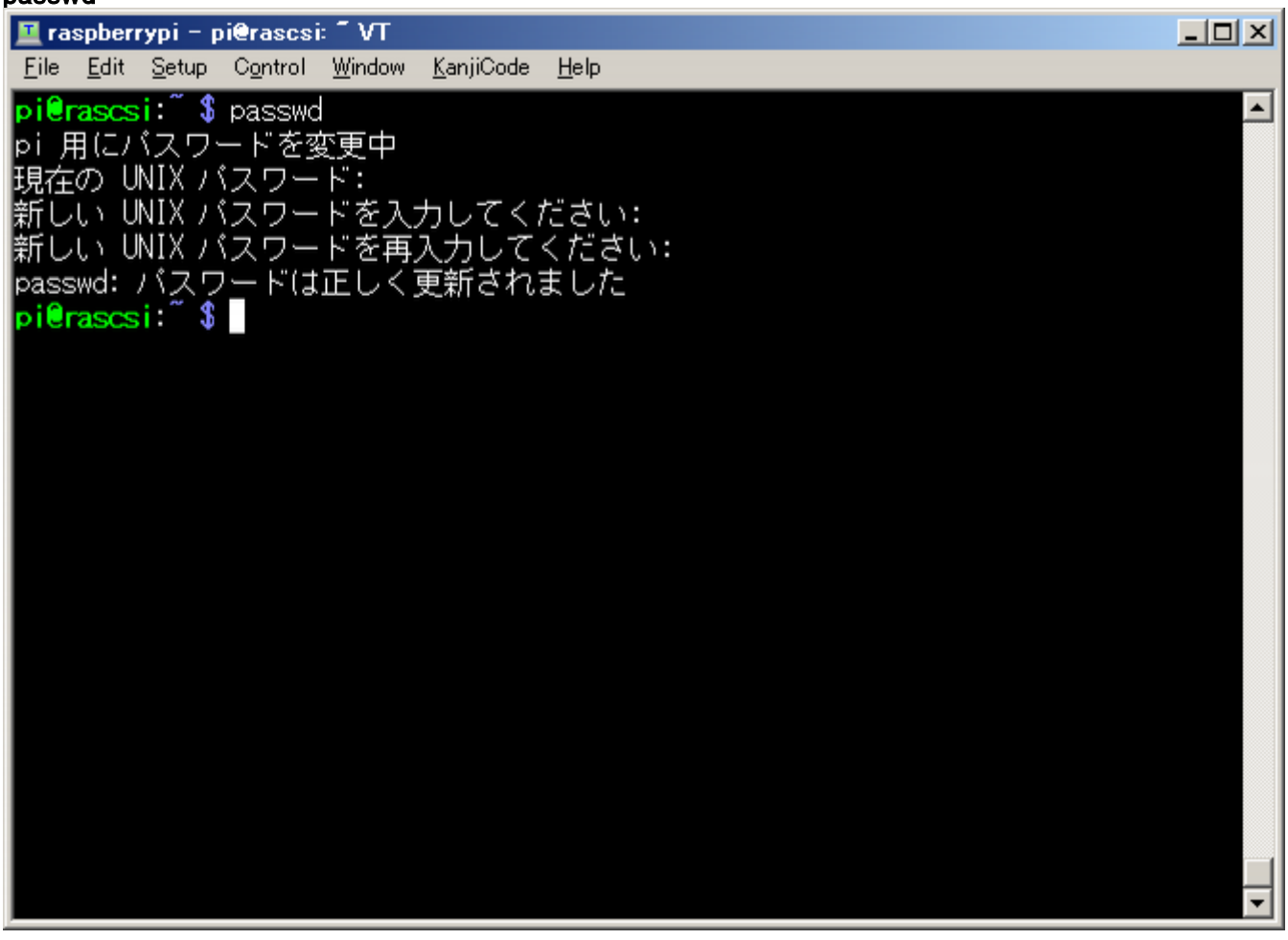
その後に、ホスト設定の一部のファイルを書き換える。書き換えなくても動くが、何かプログラムを実行するたびに
「ホスト (ホスト名) の名前解決ができません」と表示されるので、気分的にあまり良くないので

```
~ $ sudo nano /etc/hosts
```

末尾の[127.0.1.1　raspberrypi]という部分を新しいホスト名に書き換える→[127.0.1.1 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

なおベアメタル版 RaSCSI の場合はそのあたりを気にする必要はありません。遠慮なく電源ブチッをやっちゃってください

Zα さんの RaSCSI for PHP を入れると、いちいち SSH でログインして電源を切らなくても、web ブラウザから切ることができるようになります

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 の場合最大 16 台！)
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 4095MB、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(T98-NEXT PC-9801 エミュレータ形式) ~ $ **sudo rascsi -ID0 scsiimg0.nhd** 拡張子「nhd」、最大容量 2047MB、ディスクイメージは T98-NEXT で作成する
SCSI インターフェース搭載機(Anex86 エプソン 98 互換機エミュレータ形式) ~ $ **sudo rascsi -ID0 scsiimg0.hdi** 拡張子「hdi」、最大容量 2047MB、ディスクイメージは Anex86 で作成する
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(Raspberry PI とファイルのやり取りをする場合は必須オプション) ~ $ **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

なお、いちいち SSH でログインしてコマンドラインからディスクイメージを切り替えるのは面倒なので、そのあたりの面倒さを解消するフロントエンドをZ α さんが開発されています

https://github.com/ztto → rascsi-php

```
~ $ cd /home/pi
~ $ wget https://raw.githubusercontent.com/ztto/rascsi-php/master/rascsiphp_setup.sh
~ $ chmod 755 rascsiphp_setup.sh
~ $ sudo ./rascsiphp_setup.sh
```

基本的には上記の「RaSCSI 半自動スクリプト」とファイル名が違うだけで一緒です
ただし、あくまで「rasctl の web 版」なので、RaspberryPI の電源を落としたり再起動すると設定は元に戻ります

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

各種エミュレータのディスクイメージ作成機能でも作れます(SASI と MO 形式は XM6 TypeG オンリー)

上記、RaSCSI ベアメタル版項目の**「RaSCSI のディスクイメージを作る」**の先にあるリンクから飛んでください

有志が作成した RaSCSI で使える便利ツール

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

X68000 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入ったディスクイメージが含まれています(RASDRIVER.HDS をマウントする)

・RASCPY(DOS 汎用)
・RASDRV for DOS(DOS 汎用)
・RASDRV for 55BIOS(PC-98 専用)

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

X68000 用ファイル共有ドライバである RASDRV の DOS 版

・RASCPY(DOS 汎用)

RASCPY は最初にリリースされた RASDRV 互換ファイル共有プログラムで、造りとしては DOS 版 FTP クライアントに近い。後に発表された RASDRV for DOS(98)版と比べると単機能だが、バッチ処理を行う際の中間プログラムとして使用する分にはまだまだ使い道がある

・RASDRV for DOS(DOS 汎用)

X68000 版「RASDRV」の DOS 版(DOS が動いてるマシンなら汎用)。Raspberry PI との接続に論理ドライブがマウントできるようになった、DOS で行える操作全てが適用できる(ただしディスクを直接触るようなのは不可)

この 2 つは、使用するのに ASPI ドライバが必要(PC-98 で使う場合フリーソフトで対応可能)

ASPI ドライバさえ用意できれば、DOS が動いているマシン全般で動く(98 でも DOS/V でも FM TOWNS でも?)

なお、ASPI ドライバの制限(?)により RaspberryPI 側のファイル/ディレクトリ名が 8 文字を超えるとファイル表示してもそのファイル/ディレクトリだけ見えなくなるが、実用上は問題ない

・RASDRV for 55BIOS(PC-98 専用)

RASDRV for DOS の PC-98 専用版。何が専用版かというと、PC-98 の SCSI ボードである「PC-9801-55」の SCSI-BIOS に依存して動作するから。55BIOS 版というからには、55 系ボードしか使えないと思われがちだが後継の PC-9801-92 や PC-9801-100、サードパーティーのCバス SCSI ボードでも BIOS 的には 55 と下位互換が保たれているので使用可能
(一部相性などの問題で使用できないボードがあるかもしれない、その場合は ASPI 版で)

55BIOS 版は ASPI ドライバを必要とせず、CONFIG.SYS に LASTDRIVE=Z と書いてプログラムを実行するだけで動作する

こちらも Raspberry PI 側のファイル名が 8 文字を超えるとそのファイルが見えなくなる。ということは ASPI の問題ではなく、DOS のファイルシステムとか DISK-BIOS の問題?

・RaSCSI for php

zα2 さん(Twitter:@z_alpha2)
https://github.com/ztto

RaSCSI/FDX68 のイメージを web ブラウザ上から切り替えるフロントエンドスクリプト

動作イメージはこんな感じ→http://projectmps.net/images/emu/za_php.jpg

導入方法などは上のほうで紹介済みです。これがあれば今まで 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。ただし rasctl の web 版なので、RaspberryPI の電源を落としたり再起動すると設定は元に戻ります。戻らないようにするには従来どおり、SSH でログインして手作業で直接設定ファイルを書き換えてください

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」(2019 年 3 月 11 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

Project M.P.S のサポートサイトのほうで、RaSCSI の応用法(主に PC-9800 シリーズ中心)を取り扱ったコンテンツを展開しています。上記のサポートサイトにアクセスし→<RaSCSI/FDX68>というメニューから入れます

筆者が「ここは分かりにくいから補足説明しておこう」「RaSCSI で〇〇を実現するにはこうすればいい」と気づいたことを、分かる範囲で記事にしていきたいと思います(どうしても機種特有の「テク」とかありますので全部は無理ですが)

RaSCSI は本当に面白いテクノロジーなので、日本のみならず世界に発信していきたいところですが、残念ながら国内でさえあまり知名度がありません。なのでこのドキュメントが少しでも普及の手助けになれば幸いです