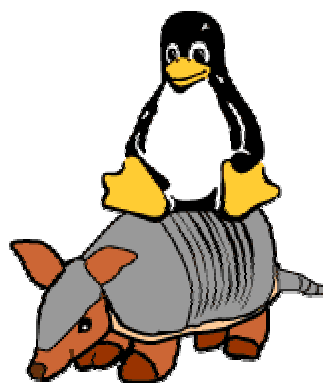




Howto's



梅沢無線電機株式会社

<http://www.umezawa.co.jp>

株式会社アットマークテクノ

<http://www.atmark-techno.com>

Armadillo 公式サイト

<http://armadillo.atmark-techno.com>

ArmadilloHowto's

Armadillo の基本的な使い方を紹介します。

• 電源を入れる前に	2
• 起動方法	5
• ログアウト・終了	6
• ネットワークに接続する	7
• telnet でログインする	8
• ftp でファイルを送受信する	11
• web ブラウザで閲覧する	14
• PC でクロス開発をする	15
• プログラムのコンパイル方法	18
• カーネルを再構築してみる	19
• ユーザーランドをカスタマイズする	22
• オンボードフラッシュメモリに書き込む	26
• Cygwin 上でクロス開発する	29
• GDB Server を使う	31

この Howto に関する情報は、

Armadillo 公式サイト
<http://armadillo.atmark-techno.com>

Howto's コーナー
2002 年 3 月 15 日 現在のものです。

最新情報は、上記の URL からご覧ください。

電源を入れる前に

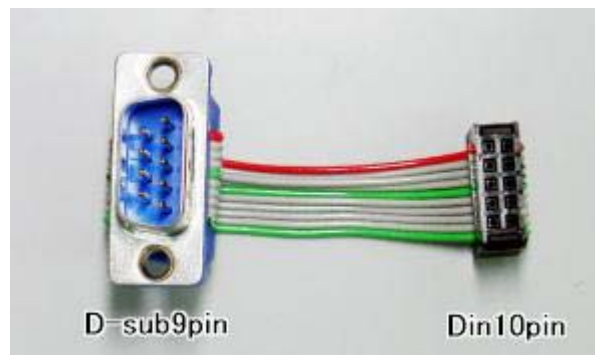
1.コネクタ接続

シリアル端子:

Armadillo は、 端末となるホスト PC とシリアルポート(COM1)で接続します。

Armadillo は CON3 がシリアルポート(COM1)となっています。 Armadillo のコネクタ形状が Din10pin となっているため、 D-sub 9pin<->Din 10pin 変換コネクタを用意します。

PC 側との接続にはシリアルクロス(リバース)ケーブルを使用します。



(D-sub 9pin<->Din 10pin 変換コネクタ)

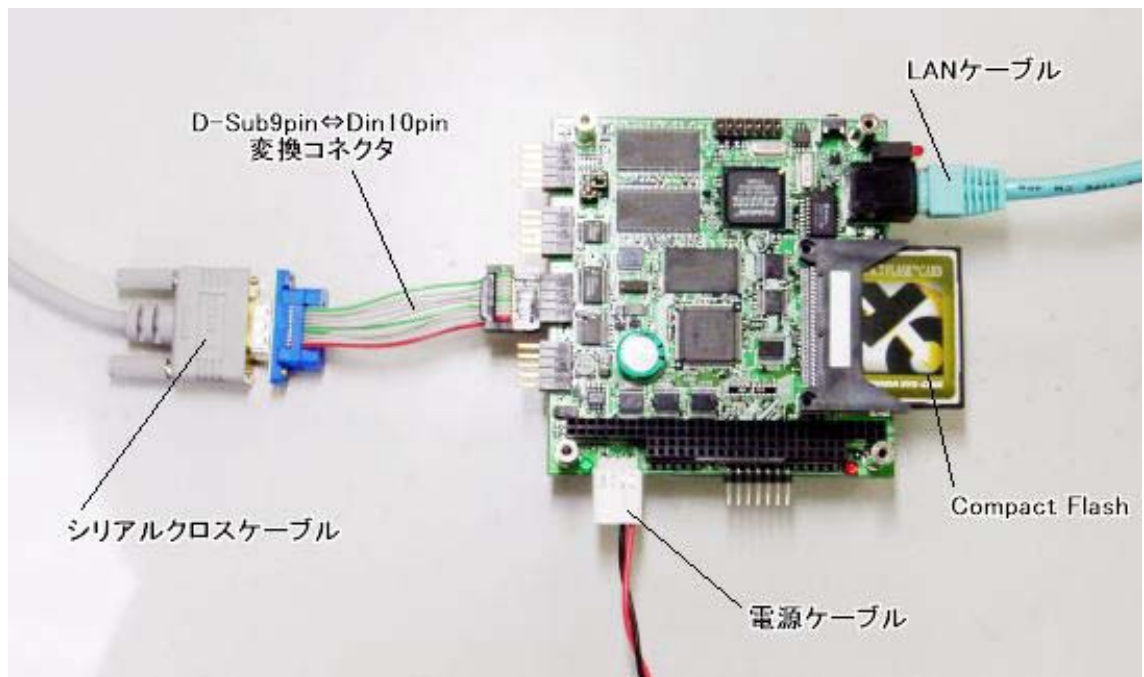
電源:

Armadillo 本体に必要な電源は、Typ.で 5V200mA です。それに見合った電源を用意する必要があります。 Armadillo 側には電源から 4 端子出ていますが、 4 番ピンに +5V、 3 番ピンもしくは 2 番ピンに Gnd を接続します。

LAN ケーブル:

ネットワーク接続を行いたい場合は、 LAN ケーブルを接続します。

CompactFlash を使用する場合は、 あらかじめ電源を入れる前にソケットに差し込みます。



(Armadillo の接続例)

2.起動モードの設定

Armadillo はジャンパピンで、起動モードを切り替えることができます。



・オンボード Flash メモリのカーネルから起動する

JP1:OFF, JP2:OFF にします。

・CompactFlash のカーネルを実行する

JP1:ON, JP2:OFF にします。

・CPU オンチップブート ROM から起動する

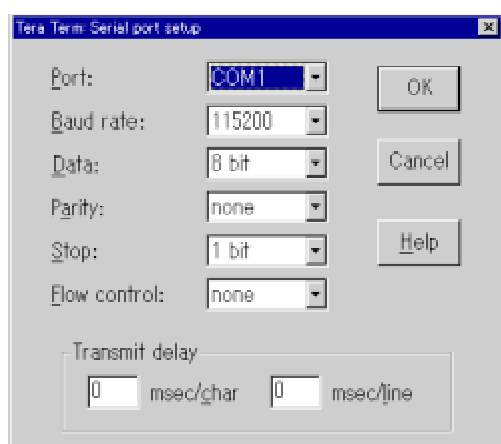
JP1:(どちらでもよい), JP2:ON にします。

3. 端末アプリケーションのインストール・設定

PC を端末として使用する場合、あらかじめホスト PC にシリアル端末アプリケーションをインストールしておきます。

ホスト PC が Windows なら、Tera Term Pro など、Linux なら、uucp の cu などを使用することができるでしょう。

シリアルポートの接続パラメータは次のように設定します。



(Tera Term Pro での設定画面)

転送レート	: 115200bps
データ長	: 8bit
パリティ	: なし
ストップビット	: 1bit
フロー制御	: なし

これで、Armadillo 起動前の準備が整いました。

[\[目次へ\]](#)

起動方法

電源ケーブル、シリアルクロスケーブルを接続し、ホスト PC のシリアル端末アプリケーションを起動したら、Armadillo の電源を ON にします。

電源スイッチを ON にすると、起動ログがコンソールに出力されます。

Armadillo をネットワークに接続していない場合や、ネットワークに接続していても DHCP を使用できない場合、「Starting DHCP for interface eth0:」と表示された後、一定時間起動スクリプトが停止します。
この場合、シリアル端末アプリケーション上で、**[Ctrl] + C** でキャンセルし、先へ進みます。

「armadillo login:」と表示されたら、ログインしてみましょう。

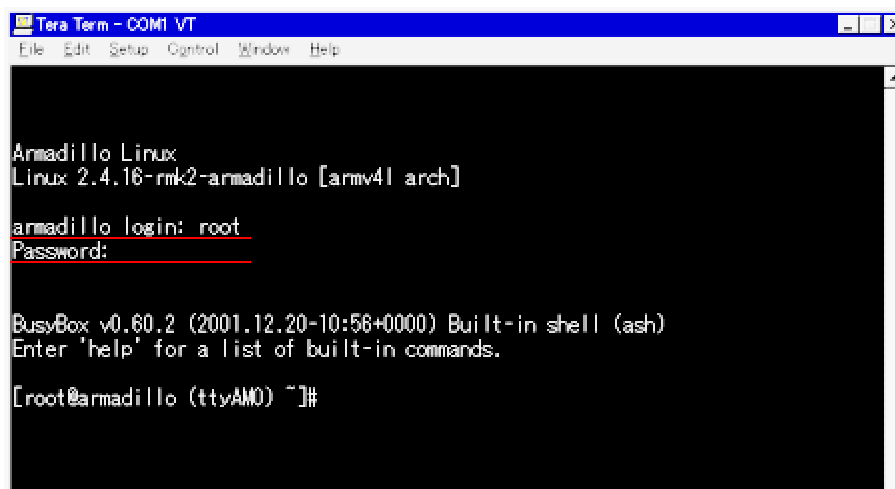
Armadillo のデフォルトでは、

ユーザ名: **root**
パスワード: **root**

もしくは、

ユーザ名: **guest**
パスワード: なし

でログインすることができます。



(オンボード Flash メモリから起動したときの root ユーザからのログイン)

[\[目次へ\]](#)

ログアウト・終了

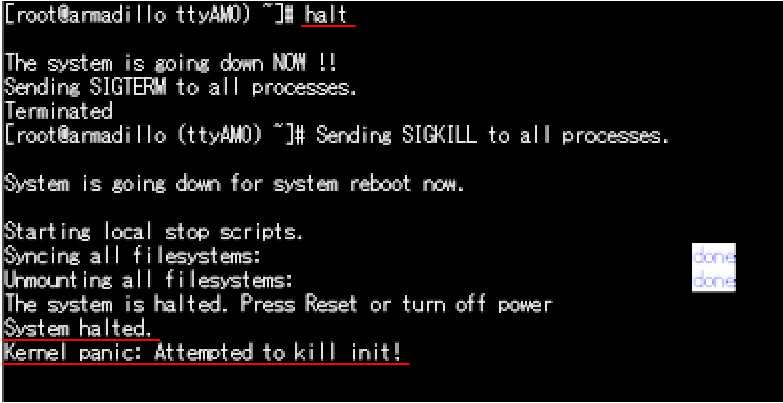
Armadillo は、**exit** コマンドでログアウトします。(logout コマンドは使えません)

終了は、**halt** コマンドを実行します。
(guest 等、一般ユーザでログインしている場合は、halt する前に、su で root ユーザになる必要があります)

System halted.

Kernel panic: Attempted to kill init!

と表示されたら、電源を OFF にします。



```
[root@armadillo ttyAM0] ~]# halt
The system is going down NOW !!
Sending SIGTERM to all processes.
Terminated
[root@armadillo (ttyAM0) ~]# Sending SIGKILL to all processes.
System is going down for system reboot now.
Starting local stop scripts.
Syncing all filesystems:
Unmounting all filesystems:
The system is halted. Press Reset or turn off power
System halted.
Kernel panic: Attempted to kill init!
```

(halt コマンドによるログアウト)

ただし、外部に機器を接続しておらず、CompactFlash をマウントしていない場合は、halt コマンドを使わずに電源を OFF にしてもよいです。

[\[目次へ\]](#)

ネットワークに接続する

Armadillo を LAN ケーブルで接続してネットワークに接続してみましょう。

DHCP サーバから IP アドレスを取得する:

DHCP が使用できる場合、起動時に自動的に DHCP サーバから IP アドレスを取得して、ネットワークに接続されています。

Armadillo に割り当てられた IP アドレスは、`ifconfig` コマンドで知ることができます。

`ifconfig eth0` と入力すると、次のようなメッセージが表示されます。

```
[root@armadillo (ttyAM0) ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:04:14:40:00:44
          inet addr:192.168.10.249  Bcast:192.168.10.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MTU:1500 Metric:1
          RX packets:11 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:1513 (1.4 kb) TX bytes:0 (0.0 b)
          Interrupt:7 Base address:0x300
```

(DHCP サーバから 192.168.10.249 という IP アドレスを割り当てられた例)

固定 IP アドレスを割り振る:

- ・root ユーザでログインします。
- ・`vi` で、`/etc/network.d/interface.eth0` を書き換えます。

IP アドレスを割り振るサンプルとして、`/etc/network.d/sample` というファイルがあるので、コピーして使うことができます。

`/etc/network.d/sample` を `interface.eth0` にコピーして必要な部分を書き換えます。


```
Tera Term - COM1 VT
File Edit Setup Control Window Help
[root@armadillo (ttyAMA0) ~]# cat /etc/network.d/interface.eth0
# network interface configuration file sample
# change settings and rename to interface.devname, i.e. interface.eth0

# network device name
INTERFACE="eth0"

# set to "yes" to use DHCP instead of the settings below
DHCP="no"

# interface settings

# IP address
IPADDRESS="192.168.10.38"      Armadilloに割り当てるIPアドレス

# netmask
NETMASK="255.255.255.0"      Armadilloを接続するネットワークのネットマスク

# broadcast address
BROADCAST="192.168.10.255"    ネットワークのブロードキャスト

# gateway address
GATEWAY="192.168.10.1"        ネットワークのゲートウェイアドレス
[root@armadillo (ttyAMA0) ~]#
```

(/etc/network.d/sample を利用して、IP アドレスを「192.168.10.69」に設定した例)

・DNS サーバを設定する場合は、vi で/etc/resolv.conf を次のように書き換えます。

nameserver [DNS サーバの IP アドレス]

・/etc/rc.d/rc.start/rc.40.network を起動します。

```
[root@armadillo (ttyAMA0) /etc/rc.d/rc.start]# ./rc.40.network
Setting up interface eth0:
Add a route to interface eth0:
done
done
```

もう一度、ifconfig eth0 で、IP アドレスが変更しているかどうかを確認することができます。

```
[root@armadillo (ttyAMA0) /etc/rc.d/rc.start]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 08:00:00:08:00:08
          inet addr:192.168.10.69  Bcast:192.168.10.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MTU:1500 Metric:1
          RX packets:17 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:3102 (3.0 kb) TX bytes:0 (0.0 b)
          Interrupt:7 Base address:0x300
```

(ifconfig コマンドで IP アドレスが「192.168.10.69」に変更されたことを確認)

ただし、オンボード Flash カーネル起動モードの場合、この方法で IP アドレスを割り振っても、電源を切ると書き換えられた情報を保存できないため、毎回設定を行う必要があります。

設定を保存したい場合は、[ユーザーランドを Linux 環境の PC で更新](#)し、シリアル経由でダウンロードしてオンボード Flash メモリを書き換えます。

[\[目次へ\]](#)

telnet でログインする

Armadillo がネットワークに接続されているなら、同じネットワークに接続された他の PC から **telnet** によりログインして操作してみましょう。

telnet でログインするには、

\$ **telnet** [Armadillo の IP アドレス]

デフォルトでは、

ユーザ名: **guest**

パスワード: なし

でログインすることができます。

(root ユーザでログインすることはできません。root 権限が必要な操作を telnet 経由で行う場合、一般ユーザでログイン後、**su** コマンドで root ユーザになります。)

```
[pooch@pc-armadillo pooch]$ telnet 192.168.10.69
Trying 192.168.10.69...
Connected to 192.168.10.69.
Escape character is '^]'.
Armadillo Linux
Linux 2.4.16-rmk2-armadillo [armv4l arch]

armadillo login: guest          guestでログイン

BusyBox v0.60.2 (2001.12.20-10:56+0000) Built-in shell (ash)
Enter 'help' for a list of built-in commands.

[guest@armadillo (ttty1) ~]$ exit    exitで終了
Connection closed by foreign host.
[pooch@pc-armadillo pooch]$
```

(telnet で IP アドレスが「192.168.10.69」の Armadillo にログインしたところ)

exit コマンドで telnet の接続を切断します。

[\[目次へ\]](#)

ftp でファイルを送受信する

Armadillo がネットワークに接続されているなら、同じネットワークに接続された他の PC から **ftp** によりログインしてファイルを送受信してみましょう。

ftp でログインするには、

\$ **ftp [Armadillo の IP アドレス]**

デフォルトでは、

ユーザ名: **anonymous**

パスワード: なし

ユーザ名: **ftp**

パスワード: なし

でログインすることができます。

```
[pooh@pc-armadillo pooh]$ ftp 192.168.10.69
Connected to 192.168.10.69.
220 armadillo FTP server (Version 6.2/OpenBSD/Linux-0.10) ready.
Name (192.168.10.69:pooh): ftp      ftpユーザでログイン
331 Guest login ok, type your name as password.
Password:                        パスワードは入力しない
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
ftp>
ftp> ls                          ログインしたディレクトリのファイル一覧
200 PORT command successful.
150 Opening ASCII mode data connection for '/bin/ls'.
dr-x--x--x  2 0      0      1024 Dec 20 11:48 bin
dr-x--x--x  2 0      0      1024 Dec 20 11:48 etc
dr-x--x--x  2 0      0      1024 Dec 20 11:48 lib
drwxrwxrwx  2 0      0      1024 Dec 20 10:30 pub
226 Transfer complete.
ftp>                             /pubディレクトリは
                                   アップロード/ダウンロードともに可能
```

(ftp で IP アドレスが「192.168.10.69」の Armadillo にログインしたところ)

ftp でアップロード:

では、ファイルを PC から Armadillo にアップロードしてみましょう。

上の図で示されているように、デフォルトでは、ログイン後 **/pub** ディレクトリがアップロード/ダウンロード可能な状態になっています。

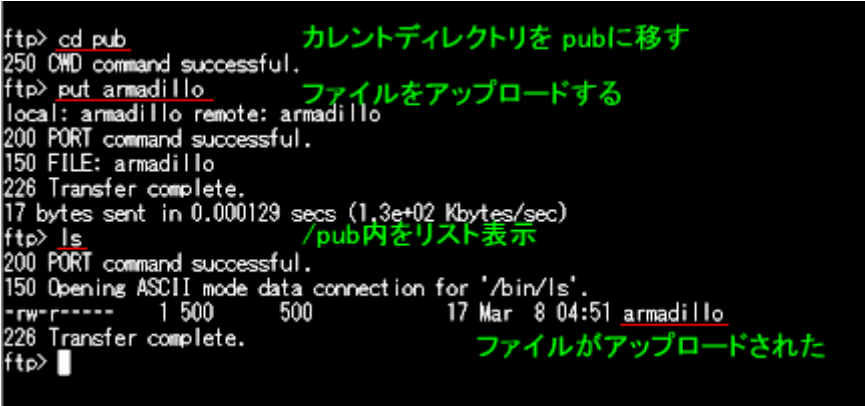
ここに、接続している PC にあらかじめ作っておいた、「armadillo」という名前のファイルを転送してみましょう。

まず、カレントディレクトリを/pub に移します。

```
ftp> cd pub
```

「armadillo」というファイルをアップロードします。

```
ftp> put armadillo
```



```
ftp> cd pub                                カレントディレクトリを pubに移す
250 CWD command successful.
ftp> put armadillo                          ファイルをアップロードする
local: armadillo remote: armadillo
200 PORT command successful.
150 FILE: armadillo
226 Transfer complete.
17 bytes sent in 0.000129 secs (1.3e+02 Kbytes/sec)
ftp> ls                                    /pub内をリスト表示
200 PORT command successful.
150 Opening ASCII mode data connection for '/bin/ls'.
-rw-r-----  1 500      500      17 Mar  8 04:51 armadillo
226 Transfer complete.                      ファイルがアップロードされた
ftp> █
```

(ftp で PC から Armadillo にファイルをアップロードしたところ)

bye もしくは **exit** で、ftp での接続を終了します。

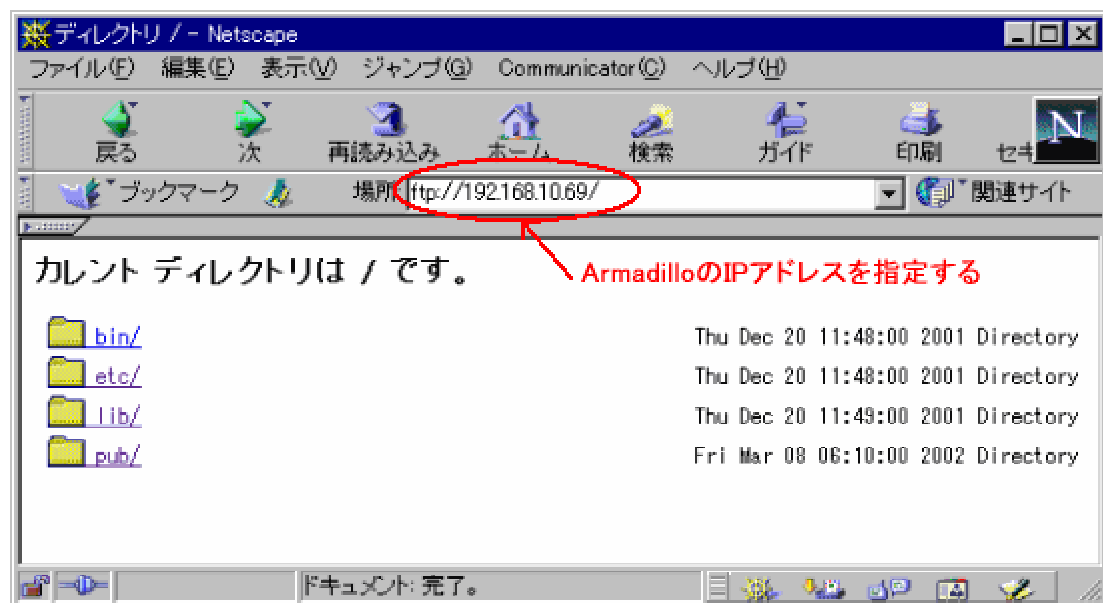
web ブラウザから ftp でファイルの送受信をする:

ftp のファイル転送は、web ブラウザ(Netscape, InternetExplorer など)からも行うことができます。

web ブラウザのアドレスバーに、

[ftp://\[ArmadilloIP アドレス\]](ftp://[ArmadilloIP アドレス])

と入力します。



(Netscape Navigator から ftp ディレクトリにアクセスしたところ)

ファイルの転送は、ドラック & ドロップなどで行うことができます。

[\[目次へ\]](#)

web ブラウザで閲覧する

Armadillo と同じネットワークに接続されている他の PC は、Armadillo 内の web ページをブラウザから閲覧することができます。

web ブラウザのアドレスバーに、

[http://\[ArmadilloIP アドレス\]](http://[ArmadilloIP アドレス])

と入力します。



(web ブラウザから Armadillo の web ページを閲覧したところ)

デフォルトでは、[/home/www-data](#) ディレクトリが、web サーバのトップディレクトリになっています。

上図のページは、[/home/www-data](#) にある、index.html を表示したものです。

この、[/home/www-data](#) ディレクトリに自分で作ったオリジナルの web ページを ftp で転送すれば、ウェブサーバになるわけです。

[\[目次へ\]](#)

PC でクロス開発をする

Armadillo は、Linux 環境の PC とシリアル接続や LAN に接続することによって、カーネルやアプリケーションのクロス開発をすることができます。

1.Linux のインストールされた PC を用意する

Debian, RedHat, Vine などの一般的な Linux ディストリビューションを、使用することができます。

Linux の PC へのインストール・設定については、それらディストリビューションのマニュアルや、参考書をご覧ください。

2.クロス開発環境パッケージをインストールする

LinuxPC にクロス開発環境をインストールしましょう。

パッケージのインストールをする前に、**su** で root ユーザになります。

インストールするパッケージ:

Linux 開発用キット付属の CD-ROM の **/cross-dev/devel** ディレクトリ下に、**deb**(Debian 系), **rpm**(RedHat 系), **tgz**(Slackware など)の 3 種類の形式でクロス開発環境が用意されています。クロス開発を行う PC でお使いのディストリビューションに合ったものを選びます。

各形式のディレクトリにある、以下のパッケージをインストールします。

binutils	Binary utilities
cpp	The GNU C preprocessor
gcc	The GNU C compiler
g++	The GNU C++ compiler
libstdc++	GNU stdc++ library
libstdc++-dev	GNU stdc++ library (development)

パッケージのインストール方法:

deb パッケージ(Debian 系)

```
> dpkg -i ***.deb
```

rpm パッケージ(RedHat 系)

```
> rpm -i ***.rpm
```

tgz 圧縮ファイル(Slackware など)

```
> cd /
```

```
> tar -xzf ***.tgz
```

(***はファイル名)

3.クロス開発環境用ライブラリをインストールする

クロス開発環境用の標準 C ライブラリのパッケージは、CD-ROM の [/cross-dev/lib](#) ディレクトリにあります。

以下のパッケージが、deb/rpm/tgz の 3 種類の形式で用意されています。

libc6-arm-cross	GNU C Library
-----------------	---------------

libc6-dev-arm-cross	GNU C Library (Development)
---------------------	-----------------------------

パッケージのインストールをする前に、[su](#) で root ユーザになります。

クロス開発を行う PC でお使いのディストリビューションに合ったものをインストールします。---> [パッケージのインストール方法](#)

4. シリアルダウンローダ/オンボードフラッシュライタをインストールする

Armadillo のオンボード Flash メモリをシリアルポート経由で書き換えるために、Linux PC にシリアルダウンローダ/オンボードフラッシュライタをインストールします。

CD-ROM の [/cross-dev/bootloader](#) ディレクトリに以下のパッケージが、deb/rpm/tgz の 3 種類の形式で用意されています。

shoehorn	CPU オンチップブート ROM と協調動作するダウンローダ
hermit	Armadillo ブートプログラムと協調動作するダウンローダ (Armadillo ブートプログラム自体も含みます)

パッケージのインストールをする前に、[su](#) で root ユーザになります。

クロス開発を行う PC でお使いのディストリビューションに合ったものをインストールします。---> [パッケージのインストール方法](#)

以上で、LinuxPC でのクロス開発を行う環境ができました。

[\[目次へ\]](#)

プログラムのコンパイル

クロス開発用の gcc をインストールしたら、PC でプログラムを作成してコンパイルしてみましょう。

ここでは、「Hello World!」と表示する、簡単なプログラムを作成します。

```
#include <stdio.h>

int main(void)
{
    printf ("Hello World! \n");

    return 0;
}
```

これを、hello.c というファイルとして保存し、次のようにコンパイルします。

```
$ arm-linux-gcc -I/usr/arm-linux/include -L/usr/arm-linux/lib hello.c -o hello
```

インクルードパス ライブラリパス ソースファイル 実行ファイル

ここでできた実行ファイル hello を、Armadillo で動かしてみましょう。
ftp で/pub ディレクトリにファイル転送したあと、

```
cd /home/ftp/pub
```

```
chmod +x hello    (実行権を与えます)
```

```
./hello
```

Armadillo で、「Hello World!」が表示されます。

[\[目次へ\]](#)

カーネルを再構築してみる

Armadillo のカーネルのソースファイルを PC でコンパイルして、カーネルイメージを作成してみましょう。

カーネルイメージの作成の前に、LinuxPC に、クロス開発環境がインストールされていなければなりません。-----[PC でクロス開発をする](#)参照

1.カーネルのソースファイルを make するドライブにコピー/展開します。

カーネルのソースファイルは、CD-ROM の `/kernel/source` ディレクトリに `tgz` 形式ファイルに圧縮されています。

ソースファイルのコピー:

```
> cp [コピー元ディレクトリ] [コピー先ディレクトリ]
```

展開:

```
> tar -xzf [カーネル圧縮ファイル]
```

2.コンフィグの設定をします。

まず、カーネルソースファイルをコピーしたディレクトリに移ります。

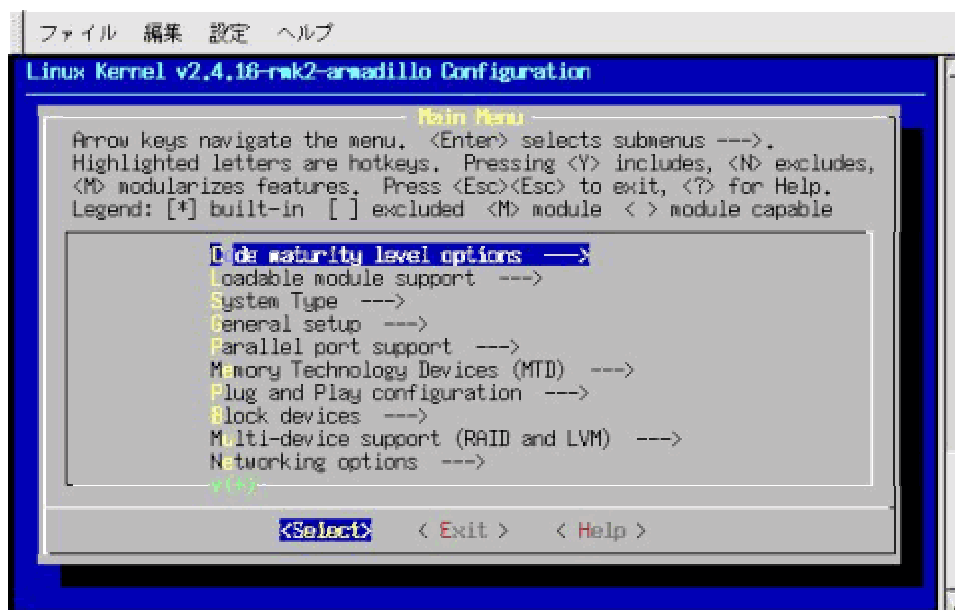
```
> cd [コピー先ディレクトリ]
```

このディレクトリに、`make` ファイルがあります。

コンフィグ設定は、

```
> make menuconfig
```

下図のようなコンフィグ設定画面になります。



(コンフィグ設定画面)

<Exit>を選択して終了したら、



と表示されるので、<Yes>を選択して設定変更を保存します。

```
Saving your kernel configuration...

*** End of Linux kernel configuration.
*** Check the top-level Makefile for additional configuration.
*** Next, you must run 'make dep'.
```

3.依存関係記述ファイルを更新します。

次に、依存関係記述ファイルを更新します。

> **make dep**

4.カーネルを make します。

カーネルを make し、Image ファイルを出力します。

```
> make r
```

これで、カーネルの作成は完了です。

カーネルソースディレクトリにできた、「Image」がカーネルイメージファイル、「Image.gz」がカーネルイメージの圧縮ファイルです。

前回の make による中間ファイルの消去方法：

```
> make clean
```

コンフィグ設定情報と、依存関係記述ファイルは消去されずにそのまま残ります。

[\[目次へ\]](#)

ユーザーランドをカスタマイズする

「ユーザーランド」とは？

Linux を起動するのに必要なカーネル以外の部分、つまり、アプリケーションやデバイスノード (デバイスドライバの入口) などのことを「ユーザーランド」と呼びます。

オンボード Flash カーネル起動モードのときは、オンボード Flash メモリから RAM Disk にユーザーランドが展開されます。

Armadillo のユーザーランドをカスタマイズしたい場合は、

1. LinuxPC 上でユーザーランド RAM ディスクイメージを更新し、
2. オンボード Flash メモリに書き込む

ことになります。

では、まずユーザーランド RAM ディスクイメージを更新してみましょう。

CD-ROM には、オリジナルディストリビューション Armadillo Linux のユーザーランド RAM ディスクイメージとして、[/armadillo-linux/initrd](#) ディレクトリに、[initrd.img.gz](#) というファイルがあります。

このファイルをクロス開発環境のインストールした PC の適当なディレクトリにコピーしておきます。

1. gzip 圧縮を解凍する

イメージファイルは gzip 圧縮されているので、解凍します。
> [gunzip initrd.img.gz](#)

ここで解凍した [initrd.img](#) が RAM ディスクイメージです。

2.RAM ディスクイメージファイルをマウントする

まず、マウントするための空のディレクトリを作成します。

> **mkdir [マウントディレクトリ名]**

このディレクトリに RAM ディスクイメージファイルをマウントします。

> **mount -o loop initrd.img [マウントディレクトリ名]**

マウントすると、マウントディレクトリの中が、Armadillo Linux のディレクトリツリーになります。

```
pc-armadillo:/home/pooh# ls
initrd.img.gz
pc-armadillo:/home/pooh# gunzip initrd.img.gz  gzip圧縮を解凍する
pc-armadillo:/home/pooh# ls
initrd.img 解凍したRAMディスクイメージ
pc-armadillo:/home/pooh# mkdir image 空のディレクトリを作成する
pc-armadillo:/home/pooh# ls
image initrd.img
pc-armadillo:/home/pooh# mount -o loop initrd.img image RAMディスクイメージを
pc-armadillo:/home/pooh# cd image 先に作ったディレクトリに
pc-armadillo:/home/pooh/image# ls マウントする
bin  etc  lib      lost+found  proc  sbin  usr
dev  home  linuxrc  mnt        root  tmp   var
pc-armadillo:/home/pooh/image#
```

(RAM ディスクイメージを、「image」というディレクトリにマウントしたところ)

3.ユーザーランドをカスタマイズしてみる

マウントディレクトリ内の Armadillo Linux を、ファイルを追加したり削除したり、設定ファイルを書き換えたりしてカスタマイズしてみましょう。

例 1)ホスト名を変更してみる

> **cd [マウントディレクトリ名]/etc**

> **vi HOSTNAME**


```

pc-armadillo:/home/pooh# cd image
pc-armadillo:/home/pooh/image# ls
bin  etc  lib      lost+found  proc  sbin  usr
dev  home  linuxrc  mnt         root  tmp   var
pc-armadillo:/home/pooh/image# cd etc
pc-armadillo:/home/pooh/image/etc# ls
HOSTNAME      gateways      hosts.allow   motd           profile        rpc
cron.d         group         hosts.deny     mtab           profile.d      securetty
crontab        gshadow       httpd.conf    network.d      protocols      services
firewall.conf host.conf     inetd.conf    nsswitch.conf  rc.d           shadow
fstab          hosts         inittab       passwd         resolv.conf    terminfo
pc-armadillo:/home/pooh/image/etc# vi HOSTNAME
viでHOSTNAMEを書き換える

```

(/etc 以下の設定ファイルを書き換えて Armadillo の設定を変更してみよう)

HOSTNAME に新ホスト名を記述します。

デフォルトでは、「armadillo」と記述されています。
この行を削除して、好みのホスト名を記述します。

```

pc-armadillo:/home/pooh/image/etc# cat HOSTNAME
pooh

```

(ホスト名を pooh に変更したところ)

例 2) 固定 IP を割り振ってみる

このように、マウントされた RAM ディスクイメージは、通常の LinuxPC と同じ方法でファイル操作や設定の変更をすることができるのです。

4.RAM ディスクイメージのマウントを解除する

ユーザーランドのカスタマイズが終わったら、RAM ディスクイメージのマウントを解除します。

カレントディレクトリをマウントディレクトリ以外のディレクトリに移します。

(設定ファイルの変更のために[マウントディレクトリ]/etc にいるのなら、

> **cd ../..**)

> **umount [マウントディレクトリ名]**

```

pc-armadillo:/home/pooh/image/etc# df
Filesystem            1k-blocks      Used Available Use% Mounted on
/dev/hda1              2885184      508828   2229796   19% /
/home/pooh/initrd.img      5947         4943       697   88% /home/pooh/image
pc-armadillo:/home/pooh/image/etc# cd ../../
pc-armadillo:/home/pooh# umount image
pc-armadillo:/home/pooh# df
Filesystem            1k-blocks      Used Available Use% Mounted on
/dev/hda1              2885184      508828   2229796   19% /
pc-armadillo:/home/pooh#

```

(RAM ディスクイメージのマウントを解除)

5.gzip 圧縮する

オンボード Flash メモリのユーザーランド領域は、約 2.5MB です。
 ここでカスタマイズしたユーザーランド RAM ディスクイメージ initrd.img は、約 6MB のファイルサイズがあります。
 それで、オンボード Flash メモリにユーザーランドを書き込むために、gzip 圧縮をします。

> **gzip -9 initrd.img**

```

pc-armadillo:/home/pooh# ls
image  initrd.img
pc-armadillo:/home/pooh# gzip -9 initrd.img
pc-armadillo:/home/pooh# ls
image  initrd.img.gz
pc-armadillo:/home/pooh#

```

(gzip で、initrd.img が initrd.img.gz に圧縮されたところ)

これで、ユーザーランド RAM ディスクイメージを更新することができました。

では、この新しい initrd.img.gz を、オンボード Flash メモリに書き込んでみましょう。
 --->NEXT: [オンボードフラッシュメモリに書き込む](#)

[\[目次へ\]](#)

オンボード Flash メモリに書き込む

カーネルイメージやユーザーランド RAM ディスクイメージを LinuxPC で作成したら、シリアルポート経由で Armadillo にダウンロードしてオンボード Flash メモリに書き込んでみましょう。

この作業には、シリアルダウンローダがインストールされている必要があります。

--->[シリアルダウンローダのインストール](#)参照

1.Armadillo の電源を入れる前にすること

- Armadillo の電源を ON にする前に、ホスト PC と Armadillo の COM1 をシリアルクロス (リバース) ケーブルで接続します。
- ジャンパピンを、**JP1: ON** , **JP2: OFF** に設定します。
- Compact Flash ソケットには何も挿入されていない状態にします。

2.Armadillo の電源を ON にする

LED(D9)が数秒間だけ点灯するので、消灯するのを待ちます。

3.**hermit** でイメージをダウンロード/オンボード Flash メモリに書き込む

Linux カーネルイメージを書き込む：

「[カーネルを再構築する](#)」の手順でカーネルを make すると、非圧縮の「**Image**」、圧縮された「**Image.gz**」の 2 つのカーネルイメージが生成されます。
オンボード Flash メモリのカーネル領域は 1,507,328 バイト(約 1.44MB)なので、「**Image**」が、このサイズよりも大きくなる場合は、圧縮された「**Image.gz**」を書き込みます。

> **hermit download -i [カーネルイメージファイル] -a [0x10000](#)**

```
[pooh@pc-armadillo source]$ ls
COPYING      Makefile.org  fs/           mm/
CREDITS      README       include/      net/
Documentation/ REPORTING-BUGS init/          scripts/
Image*       Rules.make   ipc/          valinux*
Image.gz     System.map   kernel/
MAINTAINERS  arch/        lib/
Makefile     drivers/     linux-2.4.16-rmk2-armadillo.tgz
[pooh@pc-armadillo source]$ hermit download -i Image -a 0x10000 ダウンロード
target: Hermit V1.3-armadillo @16:52:29, Dec  5 2001
serial: 0x00024000 (147456) bytes of 1457515... 転送状況が表示される
```

(カーネルイメージファイル「**Image**」を Armadillo のオンボード Flash メモリに書き込む)

```
serial: completed 0x00163d6b (1457515) bytes.
```

と表示されたら、カーネルイメージのダウンロード/書き込みが正常に完了します。

ユーザーランド RAM ディスクイメージを書き込む:

> **hermit download -i [ユーザーランドイメージファイル] -a 0x180000**

```
[pooh@pc-armadillo initrd]# ls -l 圧縮されたユーザーランドRAMディスクイメージ
initrd.img.gz -rw-r--r--
[pooh@pc-armadillo initrd]# hermit download -i initrd.img.gz -a 0x180000
target: Hermit V1.3-armadillo @16:52:29, Dec  5 2001 ダウンロード
serial: 0x00029000 (167936) bytes of 2044025... 転送状況が表示される
```

(ユーザーランドイメージファイル「**initrd.img.gz**」を Armadillo のオンボード Flash メモリに書き込む)

```
serial: completed 0x001f3079 (2044025) bytes.
```

と表示されたら、ユーザーランド RAM ディスクイメージのダウンロード/書き込みが正常に完了します。

4.Armadillo の電源を OFF にする

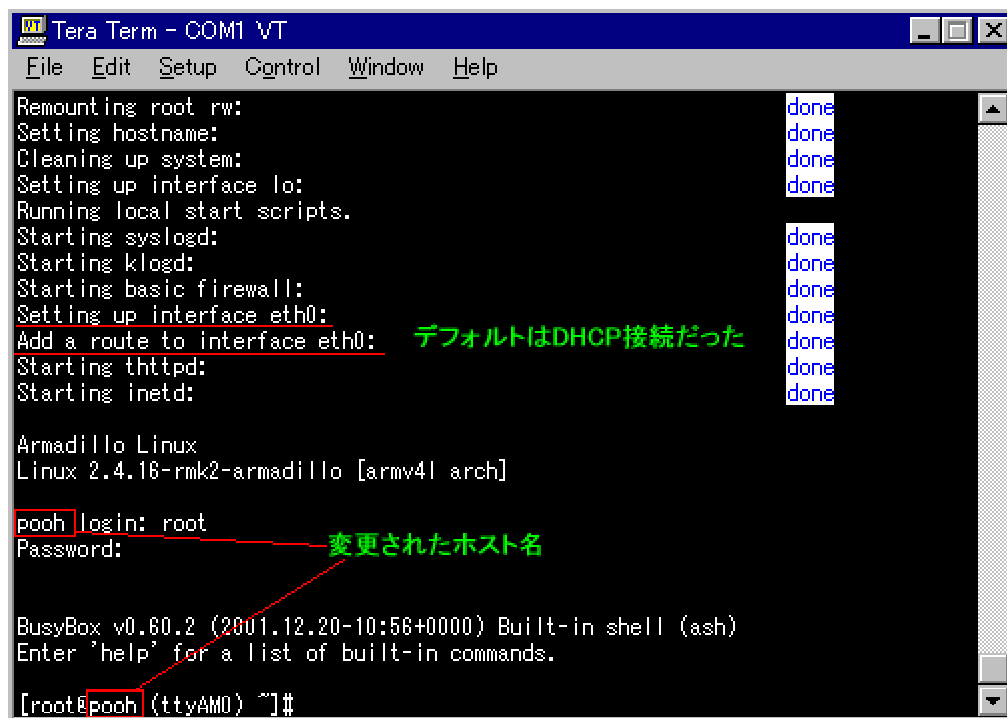
これで、オンボード Flash メモリへの書き込みは完了しました。

5.確認してみよう

では、ジャンパの設定を **JP1: OFF** , **JP2: OFF** にして、Armadillo の電源を ON にしてみましょう。

設定どおり、Armadillo が起動していますか？

もし、固定 IP アドレスを割り振る設定([固定 IP アドレスを割り振る参照](#))をしたのであれば、起動ログがかわりますし、ホスト名を変更([ホスト名を変更してみる参照](#))したのであれば、ログイン時のホスト名の表示やプロンプトが変わっていると思います。



```
Tera Term - COM1 VT
File Edit Setup Control Window Help

Remounting root rw: done
Setting hostname: done
Cleaning up system: done
Setting up interface lo: done
Running local start scripts.
Starting syslogd: done
Starting klogd: done
Starting basic firewall: done
Setting up interface eth0: done
Add a route to interface eth0: デフォルトはDHCP接続だった done
Starting thttpd: done
Starting inetd: done

Armadillo Linux
Linux 2.4.16-rmk2-armadillo [armv4l arch]

pooh login: root
Password:
BusyBox v0.60.2 (2001.12.20-10:56+0000) Built-in shell (ash)
Enter 'help' for a list of built-in commands.

[root@pooh] (ttyAMA0) ~]#
```

(Armadillo の IP アドレス、ホスト名の設定が変更された例)

[\[目次へ\]](#)

Cygwin 上でクロス開発する

Armadillo は、Cygwin を利用することによって、カーネルやアプリケーションのクロス開発を Windows 環境で行うことができます。

1.Cygwin のインストール

[Cygwin のサイト](#)からインストーラ (setup.exe) をダウンロードし、PC にインストールしてください。途中インストールするパッケージのカテゴリを選択するダイアログが表示されますが、ここではデフォルトのものに加え「Devel」「Utils」パッケージも選択しインストールしてください。他は全てデフォルトのままで結構です。

インストールが済んだら `C:\cygwin\cygwin.bat` をダブルクリックします。bash が起動するので `mount` コマンドを実行し、現在のマウント状況が以下の様になっていることを確認してください。

```
$ mount
c:\cygwin\bin on /usr/bin type user (binmode)
c:\cygwin\lib on /usr/lib type user (binmode)
c:\cygwin on / type user (binmode)
c: on /cygdrive/c type user (binmode,noumount)
```

2.クロス開発環境パッケージをインストールする

Cygwin 用のコンパイル済みのバイナリを tgz 形式で用意しました。[こちらのサイト](#)より `binutils` と `gcc` のパッケージをダウンロードし、bash から以下のように展開してください。

```
$ tar xzf arm-linux-binutils-cygwin-2.9.5.0.37.tgz -C /
$ tar xzf arm-linux-gcc-cygwin-2.95.3-0.tgz -C /
```

3.クロス開発環境用ライブラリをインストールする

クロス開発環境用の標準 C ライブラリのパッケージは Linux 用のものをそのまま利用します。[こちらのサイト](#)より `libc6-arm-cross`、`libc6-dev-arm-cross` をダウンロードし、bash から以下のように展開してください。

```
$ tar xzf libc6-arm-cross-2.1.3-19.tgz -C /  
$ tar xzf libc6-dev-arm-cross-2.1.3-19.tgz -C /
```

以上で、Cygwin 上でのクロス開発を行う環境ができました。

[\[目次へ\]](#)

GDB Server を使う

gdbserver を使うことによって、Armadillo でイーサネット経由のリモートデバッグをすることができます。

1.ARM 用 gdb と、gdbserver のダウンロード

Armadillo でクロスデバッグを行うには、ARM クロスデバッガと Armadillo 上で動く gdbserver が必要です。

- ・ [i386 用 ARM クロスデバッガ](#)
- ・ [Armadillo 上で動く gdbserver](#)

2.ダウンロードしたファイルを解凍

arm-linux-gdb.gz と gdbserver.gz は、gzip で圧縮してあります。ホストコンピュータ上で gzip を使い、解凍します。

```
$ ls
arm-linux-gdb.gz gdbserver.gz
$ gzip -d arm-linux-gdb.gz
$ gzip -d gdbserver.gz
$ ls
arm-linux-gdb gdbserver
```

3.転送

伸張した gdbserver を Armadillo に転送します。/usr/bin の下に入れておくと便利です。ftp での転送方法は「[ftp でファイルを送受信する](#)」を、ルートイメージを変更する場合は「[ユーザランドをカスタマイズする](#)」をご覧ください。転送後、ファイルパーミッションを確認してください。

4.実行

gdbserver の簡単な使い方を説明します。詳しくは gdb のマニュアル等を参照してください。

ここからは

- ・ 「arm\$」を Armadillo 上でのプロンプト
- ・ 「host\$」をホストコンピュータ上でのプロンプト
- ・ 「192.168.1.1」をホストコンピュータの ip
- ・ 「192.168.1.2」を Armadillo の ip
- ・ 「9876」をデバッグに使用するポート番号

とします。適時置き換えてください。

まず、Armadillo 上で gdbserver を起動します。

```
arm$ gdbserver 192.168.1.1:9876 /usr/bin/gdbserver  
Process /usr/bin/gdbserver created; pid = 267
```

続いて、host コンピュータから Armadillo 上の gdbserver に接続します。

```
host$ arm-linux-gdb gdbserver  
GNU gdb 5.2.1  
Copyright 2002 Free Software Foundation, Inc.  
GDB is free software, covered by the GNU General Public License, and you are  
welcome to change it and/or distribute copies of it under certain conditions.  
Type "show copying" to see the conditions.  
There is absolutely no warranty for GDB. Type "show warranty" for details.  
This GDB was configured as "--host=i686-pc-linux-gnu --target=arm-linux"..  
(gdb) target remote 102.168.10.2:9876  
0x40002740 in ?? ()
```

接続に成功している事が gdbserver でも確認できます。

```
Remote debugging from host 192.168.10.1
```

ホストコンピュータに戻り、ブレイクポイントをつけて継続実行させます。

```
(gdb) b main
```

Breakpoint 1 at 0x20019fc: file server.c, line 67.

(gdb) c

Continuing.

Breakpoint 1, main (argc=1042, argv=0x112) at server.c:67

67 server.c: No such file or directory.

in server.c

(gdb)

[\[目次へ\]](#)