「玄箱」をカスタマイズしてC言語の勉強をしよう

CygwinにPowerPC クロス・コンパイル環境を作る

岸 哲夫

● 玄箱って何？

有名なPC関連製品のブランドに「玄人志向」というものがあります．おもしろいものができるなら，はんだ付けも，OSの再インストールも，感電も（？）厭わないマニアな人に大人気のブランドです．

「玄箱」が出て数年が経ちますが，「2ちゃんねる」にはこれをネタにしたスレッドがいまだにたくさん立っているほど人気の商品です．

マニアがこのようなものを作り，自分のサイトで自慢するならともかく，コンシューマ向けにこのようなものを出すとは驚きました．しかも実験にも実用にも使えて，パフォーマンスも良いすばらしい製品です．ファイル・サーバ用にと，Celeronを探して中古のマザーボードで安いPCを組み立てるくらいなら，こちらのほうが安くて安定しています．

玄箱に関する詳しい情報は**図A**のWebサイトをご覧ください．CPUはPowerPC 200MHz（ギガビットEthernet対応の上位機では266MHz）を採用しています（**図B**）．

この製品はファイル・サーバですが，ただのファイル・サーバではありません．PowerPCをCPUに採用しているので無限の可能性を秘めています．低スペックの小型コンピュータとして十分使えるのです．KURO-BOXコンテストという催しには，マニアな人々が集まって玄箱の使い方を熱く競い合っています（**図C**）．

この玄箱が動作するバイナリ・コードは，もちろんPowerPC用にコンパイルされたものなので通常の（x86用の）Linuxでは動作しません．Apple社の古いマッキントッシュにPowerPC向けLinuxを入れたものなら動くかもしれませんが試していません．今回はCygwinにPowerPCクロス・コンパイル環境を作ってみました．

それを試す前に玄箱が正常に動作していることを確認してください（**図D**）．もちろん，Cygwinも正常に動作していますね？

● Cygwin上にクロス・コンパイル環境を構築

▶Cygwinの環境を設定

まずCygwinの環境を設定します．

自分のホーム・ディレクトリにある`.bash_profile`の最後の行に，`export PATH="$PATH:/usr/local/ppc/bin"`という行を付け加えましょう．

その後再ログインするか`source .bash_profile`を実行すれば反映されます．

必要なツールは，

```
http://www.dnsbalance.ring.gr.jp/
```

からダウンロードします．

以前，SHのクロス・コンパイル環境を作る記事ではGCC2.95を使いましたが，今回はGCC3.2を使いました．

① gcc3.2
② binutils-2.13

図A　玄人志向の公式サイト（`http://www.kuroutoshikou.com/home.html`）

図B　玄箱の情報

KEYWORD ── 玄人志向，玄箱，KURO-BOXコンテスト，Cygwin，クロス・コンパイル環境，gcc3.2，binutils-2.13，glibc-2.2.2

図C　熱い工作のサイト（http://www.kuroutoshikou.com/products/test/kuro_award3.html）

③ glibc-2.2.2

以上のツールをダウンロードします．

次にネットワークで見つけた，パワー・ユーザが作成したビルドに必要なヘッダです．

```
http://www.angel.ne.jp/~tane/prog/arc_cygwin/
                                    glibc_head.tar.gz
http://www.angel.ne.jp/~tane/prog/arc_cygwin/
                                glibc-2.2.2-patch.diff
```

以上2点のファイルを，作者に感謝しつつ上記のURLからダウンロードします．

▶binutilsの作成

まずはbinutilsの作成です．

ダウンロードしたものはすべて~/に置きます．

```
$ cd ~/
$ tar zxvf binutils-2.13.tar.gz
$ cd binutils-2.13
$ ./configure --target=powerpc-linux --prefix=/
                                        usr/local/ppc
$ make
$ make install
```

以上でbinutilsが作成されました．エラーが起きる場合は環境をチェックします．

▶GCCの作成

次にGCCの作成です．貴重なコミュニティの産物であるヘッダを使うので，両方とも解凍します．

```
$ cd /usr/local/ppc
```

ここにダウンロードしたヘッダをおきます．

```
$ tar zxvf glibc_head.tar.gz
```

ヘッダを解凍します．

```
$ cd ~/
$ tar zxvf gcc-3.2.tar.gz
$ cd gcc-3.2
```

図D　玄箱にWebブラウザでアクセスしているようす

```
$ ../configure --target=powerpc-linux
    --prefix=/usr/local/ppc --enable-languages=c
    --with-headers=/usr/local/ppc/glibc/include/
              --disable-threads --disable-shared
$ make
$ make install
$ /bin/rm -fr /usr/local/ppc/powerpc-linux/
                                        sys-include
```

これでGCCの1回目のビルドが成功しました．エラーが起きた場合，やはり環境をチェックします．

▶glibcの作成

次にglibcを作成します．

```
$ mkdir /usr/local/ppc/powerpc-linux/include
$ cp -r /usr/local/ppc/glibc/include/linux
          /usr/local/ppc/powerpc-linux/include/.
$ cp -r /usr/local/ppc/glibc/include/asm
          /usr/local/ppc/powerpc-linux/include/.
$ cd ~/
$ tar zxvf glibc-2.2.2.tar.gz
$ cd glibc-2.2.2
$ patch -p1 < ~/buildtools/
                            glibc-2.2.2-patch.diff
$ AR=powerpc-linux-ar CC=powerpc-linux
    -gcc RANLIB=powerpc-linux-ranlib ./configure
   --host=powerpc-linux --prefix=/usr/local/ppc/
    powerpc-linux --with-headers=/usr/local/ppc/
   powerpc-linux/include/--disable-sanity-checks
$ make gnu/lib-names.h
$ R=powerpc-linux-ar CC=powerpc-linux-gcc RANLIB
      =powerpc-linux-ranlib ../configure --host=
   powerpc-linux --prefix=/usr/local/ppc/powerpc
    -linux --with-headers=/usr/local/ppc/powerpc
         -linux/include/ --disable-sanity-checks
              --disable-shared --disable-profile
$ make
$ make install
```