第2章 玄箱PRO活用の第一歩

Debian環境を整えよう

第1章で，玄箱PROでDebianが動作するようになりました．まだ環境が整っていないので，第2章では環境を整えていきます．まず，自分でプログラムをコンパイルできるように開発環境を用意し，locale，NTPを使った時刻合わせを設定します．

また，USBフラッシュ・メモリを使えるようにし，USBフラッシュ・メモリから起動することで，不要なときにはHDDの回転を止める静音環境を作ってみます．そして，USBシリアル・ポートを使えるようにし，メール環境を整えます．

2-1 開発環境を整える(gcc，make，patch，ライブラリ)

● 開発環境を整える(apt-get install)

インストールした直後は，gccなどの開発環境はインストールされていないので，aptで追加します．

```
apt-get install gcc make libc-dev libc6-dev
```

を実行します．次のようになりました．

```
# apt-get install gcc make libc-dev libc6-dev ←(入力)
Reading package lists... Done
Building dependency tree... Done
Note, selecting libc6-dev instead of libc-dev
The following extra packages will be installed:
  binutils cpp cpp-4.1 gcc-4.1 libssp0Linux-kernel-headers
Suggested packages:
  binutils-doc cpp-doc gcc-4.1-locales manpages-dev autoconf automake1.9
  libtool flex bison gdb gcc-doc gcc-4.1-doc glibc-doc make-doc-non-dfsg
Recommended packages:
  libmudflap0-dev
The following NEW packages will be installed:
  binutils cpp cpp-4.1 gcc gcc-4.1 libc6-dev libssp0Linux-kernel-headers make
0 upgraded, 9 newly installed, 0 to remove and 0 not upgraded.
Need to get 9283kB of archives.
After unpacking 33.1MB of additional disk space will be used.
Do you want to continue [Y/n]? ←(エンターを押す)
Get:1 http://ftp.jp.debian.org etch/main binutils 2.17-3 [2485kB]
Get:2 http://ftp.jp.debian.org etch/main cpp-4.1 4.1.1-21 [2006kB]
Get:3 http://ftp.jp.debian.org etch/main cpp 4:4.1.1-15 [11.6kB]
```

```
---------------------------------- <省略> -------------------------------
Setting up gcc-4.1 (4.1.1-21) ...
Setting up gcc (4.1.1-15) ...

Setting upLinux-kernel-headers (2.6.18-7) ...
Setting up libc6-dev (2.3.6.ds1-13) ...
Setting up make (3.81-2) ...
```

簡単ですね．patchコマンドもあると便利(カーネル再構築で使います)なので，同様にインストールします．以下を実行します．

```
apt-get install patch↵
```

2-2 localeのサポート

● 日本語のファイル名に対応させたい

UNIXだけで作業していると日本語のファイル名を使うことは少ないのですが，Windowsのファイルをやりとりすると，ファイル名に日本語を使いたくなることがあります．そこでlocalesをインストールして，lsなどで，日本語ファイル名が使えるようにします．

Column 2-1 クロス開発とセルフ開発

● 二つの開発の方法

プログラムを実行するマシン(ターゲット)で開発することをセルフ開発と呼び，その開発環境をセルフ開発環境と呼びます．開発用のマシンをターゲットのマシンとは別に用意する場合(とくにアーキテクチャが違うマシンの場合)には，クロス開発と呼び，開発環境をクロス開発環境と呼びます．

それぞれで使用するコンパイラは，セルフ・コンパイラ，クロス・コンパイラと呼びます．

● メリットとデメリット

セルフ開発環境のメリットは，なんといってもコンパイルが終わったら，すぐに実行ができることです．またターゲットと開発マシンで同じ名前のヘッダがある場合などに混乱することもありません．

クロス開発環境のメリットは，ターゲットが非力なときにも，開発には十分な能力のあるマシンを使うことができることです．

● 玄箱PROの開発環境としては？

Debianが基本的にはセルフ開発環境なので，Debian化した玄箱PROの開発環境としては，セルフ開発環境がapt-getだけで構築できるので便利です．その代わり，最近のパソコンと比較すると非力なCPUのため，コンパイルに時間がかかります．

パソコン上のWindowsや，Debianにクロス開発環境を構築すると，最新のCPUや十分なメモリを利用することで，コンパイル時間を短くすることができます．とくにカーネル再構築などのときには時間の大幅な短縮になります．

一方で，コンパイル済みのライブラリを利用するときには，玄箱PRO上とクロス開発環境で同じライブラリとヘッダを用意する必要があり，整合性を保つ手間がかかります．そこで，本書ではセルフ開発環境を利用しています．

● localesのインストール

localesのインストールはapt-getで行います．

```
apt-get install locales
```

● locales の設定

インストールしたら設定をします．

```
dpkg-reconfigure locales
```

説明が表示されるので，読んでリターン・キーを押します(**図2-1**)．

矢印キー(PageUp，PageDown)でスクロールしていき，ja_JP.EUC-JP，ja_JP.UTF-8に，スペース・キーでマークを入れます．マークを入れたらTabキーでOKに移動します(**図2-2**)．

デフォルトのlocaleが尋ねられます．ここでは，Noneにしました(**図2-3**)．

図2-1　localesの設定①
説明を読む．

図2-2　localesの設定②
設定するlocaleを選ぶ．

図2-3　localesの設定③
デフォルトの設定．

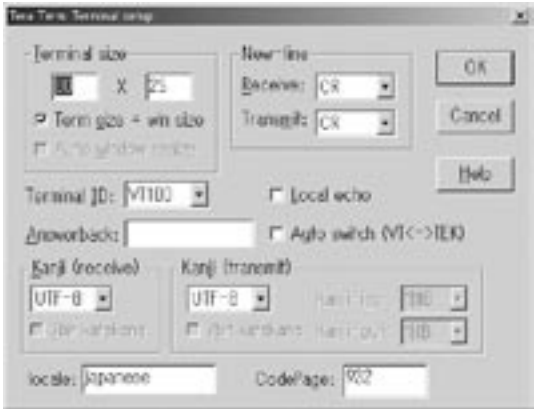

図2-4　TeraTermの設定例
TeraTermのメニューのSetup内のTerminalをクリックする．