Linuxによるセキュリティ入門（2）
―― Linuxカーネルの構築 ――

西 村 竜 一

Ⅰ．カーネルの構築してますか？

みなさんがお使いのLinuxディストリビューションは，Linuxカーネルと実際に利用するのに必要なプログラムや設定ファイル，X Windowシステムなどから構成されています。この連載でとりあげているDebianももちろん同様です。Debianパッケージに収録されているプログラムは，apt-getコマンドを利用して最新のものにアップグレードが容易にできることはこれまで説明しました。それではLinuxカーネルに関してはどうなのでしょうか？今回は，まず，Linuxカーネルのバージョンについての説明からはじめます。

現在のLinuxのカーネルには大きくわけて安定版と開発版の２種類のものがあります。バージョンが2.2.22や2.4.20のようにバージョン2.x.yのxが偶数のものが安定版，2.5.50のようにxが奇数のものが開発版です。自分が利用しているカーネルのバージョンは，

```
% uname -r
```

で調べることができます。

安定版では，バージョンアップによる変更はバグフィックスなどの小さなものがほとんどです。一般的に，Linuxシステムを利用するときは安定版のカーネルを使うことになります。一方，開発版は，次期安定版のためのテストの意味合いが強く，新しい機能や大幅な変更が頻繁に行われます。こちらは，カーネルの開発者やカーネルおっかけと呼ばれるような人が使うことになります。新しいバージョンの開発版カーネルをインストールしたら，システムが起動すらしなくなったということもしばしばありますので，利用には注意してください。Linuxの発展に貢献するには，開発版カーネルへの挑戦も良いかなとも思うのですが，本連載ではLinuxシステムの安定運用を目指してますので，やはり安定版のカーネルを使うことにします。ディストリビューションに標準でインストールされるのも安定版カーネルです。

安定版のカーネルにもバグが潜んでいるときがあります。これらの修正や機能の拡張，デバイスドライバの追加などされたとき，新しいバージョンの安定版カーネルがリリースされます。バージョンナンバは，2.x.yのyに１を足したものが使われます。この原稿執筆時点で最新の安定版カーネルは，2002年11月28日にリリースされた2.4.20です。カーネルの最新バージョンは，http://www.kernel.org/で確認できます。また，少しだけ格好良く？以下のコマンドで確認

することもできます。

```
% finger @kernel.org
```

fingerコマンドは，Debianでは`finger`パッケージに含まれていますので，必要なら`apt-get`で追加してください。

最近，Linuxのメーリングリストを見ていると，カーネルはディストリビューションをインストールしたときのものをそのまま利用するケースが増えているようです。しかし，安定版カーネルにも古いものにはセキュリティホールやバグが含まれていることがあります。特に，2.x.yのxが変わった直後の安定版カーネル（例えば2.2から2.4）には，致命的なバグが含まれていることがあります。現在の安定版カーネルの系列である2.4も，本当に安心してサーバに使えるようになったのは2.4.16の頃だった気がします（この辺の記憶はすごく曖昧です）。おそらくディストリビューションに含まれるカーネルは安定して使えるものになっているのでしょうが，新しい方がより良いことも多々あります。基幹サーバなのでどうしても再起動できない場合を除いて，カーネルを時々更新するのが良いのではないでしょうか。

また，ディストリビューションに付属のカーネルには，あらゆるシステム構成に対応するため，不必要な機能も含まれます。例えば，SCSIカードやネットワークカードなどの各種ドライバです。これら不必要な部分を削って，システムに適したスリムなカーネルを構築すると，メモリの節約になります。ということで，以下では，Debianでの具体的なカーネル構築の方法について順を追って説明しましょう。

Ⅱ．カーネル構築のためのパッケージと前準備

これまで説明の中で，余分なものは潜在的セキュリティホールになる可能性があるため，必要不可欠なパッケージしかインストールしてはならないと述べてきました。しかし，カーネルを構築するためには，C言語のコンパイラなどの以下のパッケージのインストールが必要です（環境によってはすでにパッケージがインストール済みの場合もあると思います）。

```
kernel-package, libc6-dev, gcc, make, bin86, bzip2, libdb3-dev,
libncurses-dev, fakeroot
```

これらパッケージは，比較的，重要度が高いのでインストールしておいても無駄にはならないと思いますが，他で使わないならインストールしない方が安全であることは確かです。例えば，C言語コンパイラが含まれていなければ，侵入者がシステムを破壊するプログラムをコンパイルして実行することが難しくなります。Debianは，カーネルの構築（設定やコンパイル）は，他の計算機上で行い，生成されたファイル（カーネルパッケージ）のみをターゲットのシステムに転送して，インストールすることが容易にできるよう作られています。手順は，上記のパッケージを

インストールしても構わない（ファイヤウォールの内側にあるような）別のDebianシステムを用意し，そちらでこれからする説明を実行し，最後に生成ファイルを転送するだけで基本的に変わりません。カーネルをインストールする計算機の重要度などを考慮して，そのシステム上で構築を行うか，別の計算機上で構築するかを判断してください。

それではいつものように`apt-get`を用いて前述のパッケージをインストールしましょう。操作は，以下のようになります（紙面の都合上，`apt-get`を２回にわけて実行していますが，1回でまとめて実行して構いません)。この操作はrootで実行する必要があります。

```
# apt-get install kernel-package libc6-dev gcc make
# apt-get install bin86 bzip2 libdb3-dev libncurses-dev fakeroot
```

つぎにカーネルのソースを取得します。Linuxカーネルのソースは，多くのミラーサイトにミラーリングされていますが，今回はRingサーバから，`wget`コマンドを用いて取得する方法を紹介します。必要に応じて，`wget`パッケージを`apt-get`でインストールしてから使用してください。

```
% wget http://www.ring.gr.jp/pub/linux/kernel.org/kernel/v2.4/linux-2.4.20.tar.bz2
```

この例では安定版カーネル2.4.20のソースアーカイブファイルを取得しています。この操作はrootである必要はありません。少しでもセキュリティ上好ましい操作を心がけて，一般ユーザで実行してください。取得したアーカイブファイルを適当な作業用のディレクトリに移動したのち，`tar`を使って展開します。以下の例では，ホームディレクトリの下の`kernel`という名前の作業用ディレクトリを使用します。

```
% mkdir ~/kernel
% mv linux-2.4.20.tar.bz2 ~/kernel
% cd ~/kernel
% tar xvjf linux-2.4.20.tar.bz2
```

`bzip2`で圧縮されたbz2ファイルを展開するために`tar`の実行時のオプションには，`j`を付けます。ただし，古い`tar`を利用している場合などは，この`j`オプションが使えないときがあります(woodyの`tar`は使えます)。その場合は，以下のように`tar`を実行してください。

```
% bzip2 -dc linux-2.4.20.tar.bz2 | tar xvf -
```

以上で，`linux-2.4.20`ディレクトリの下にカーネルソースを展開できたはずです。

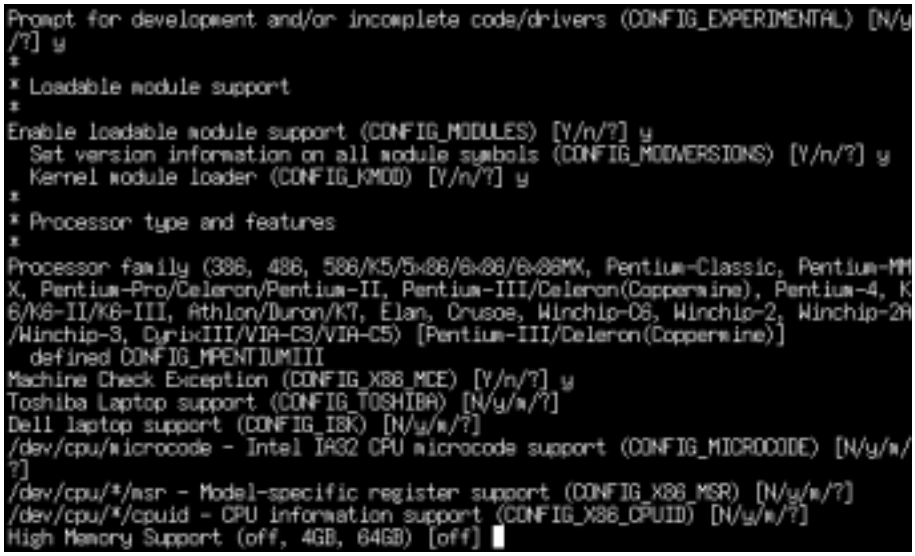

図１　make configの実行画面

Ⅲ．カーネルの設定

つぎにカーネルの設定をします。現在のLinuxカーネルのソースは，各種インタフェースカードなどのデバイスドライバやさまざまなファイルシステムなど多くの機能のコードを含めた形で配布されています。これら機能から自分が必要なものを選択して，カーネルを構築することでメモリの消費量の少ないスリムなカーネルを作ることができます。また，計算機に新しいデバイスを追加したときは，設定によりそのデバイスのドライバを追加し，カーネルを再構築する必要があります。

現在のLinuxカーネルには，３つの設定方法が用意されています。それぞれは以下のようになります。

`make config`：対話形式（図１）

Linuxカーネル当初からある対話形式での設定方法。

`make menuconfig`：キャラクタベースのメニュー形式（図２）

おそらく現在最も良く使われているメニュー形式での設定方法。

`make xconfig`：グラフィカルベースのメニュー形式（図３）

Tcl/Tkを用いたGUIベースの設定方法。使うにはX Windowシステムが動作している必要がある。

`make xconfig`は，X Windowシステムが動いている必要があり，使い勝手もあまり良くないので説明を省略します。著者が普段，カーネルの設定をするときは，まず`make config`でひととおり設定を行い，後から`make menuconfig`でもう一度，設定の確認と修正をしています。

図２　make menuconfigの実行画面

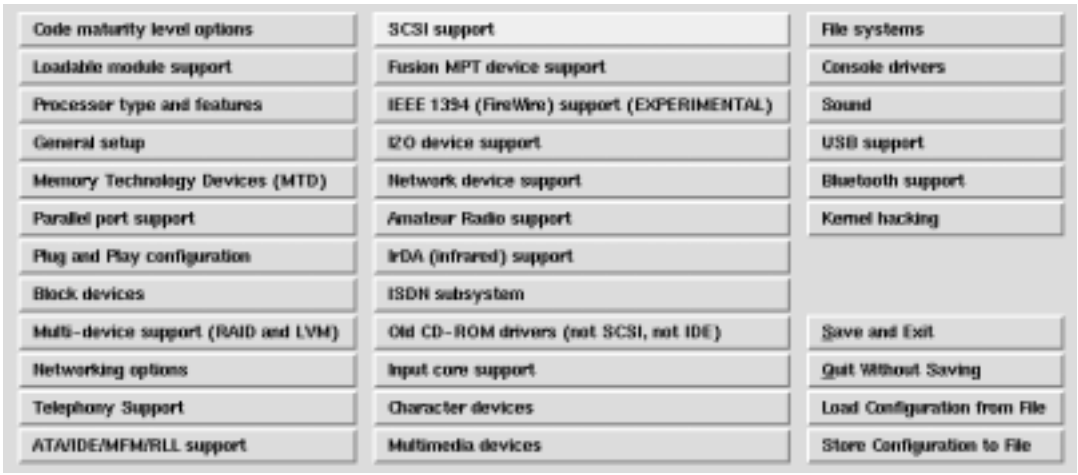

図３　make xconfigの実行画面

それでは，まず，make configを実行しましょう。先ほどカーネルソースを展開したディレクトリにカレントディレクトリを移動した後，make configを実行します。

```
% cd ~/kernel/linux-2.4.20
% make config
```

ここから項目ごとの質問に対話式に答えていきます。まず，最初に

```
Prompt for development and/or incomplete code/drivers (CONFIG_EXPERIMENTAL) [N/y/?]
```

と聞かれます。この質問にはyと答えてください。カーネルに含まれる機能の中には，現状では開発中のものも含まれます。それらを利用するか？という質問です。すこし怖いかもしれませんが，開発中のものにも十分実用的な機能やぜひ利用したい機能もあるので，Yesを選択することにします。

ここで質問の末に[N/y/?]と表示されますが，NはNo，YはYesを意味します。それぞれキーボードから，yもしくはnを入力後，リターンキーを入力してどちらかを選択します。また，最初に大文字で表示されるのがデフォルトの選択枝です。yとnのどちらも入力せずにリターンキーのみの場合，デフォルトが選択されます。最後の?はヘルプを意味します。キーボードから?を入力すると，その質問に関する説明文が表示されます。質問の意味がわからないときは参照してください（ヘルプを見ても意味がわからないことも多いですけどね）。

また，質問によっては以下の例のように選択枝の中にmが登場します。

```
Normal floppy disk support (CONFIG_BLK_DEV_FD) [Y/m/n/?]
```

この例は，フロッピーディスクのサポートをカーネルに組み込みかという項目です。もちろんフロッピーディスクを使うときはy，使わないときにはnと答えれば良いのですが，項目によっては判断できないものもあります。その場合，mはモジュール化が可能な機能なので，mと答え，とりあえずモジュール化しておくのが良いでしょう。モジュール化した機能は，システムが起動後，そのモジュールを読み込みことで有効にすることができます。なお，モジュールを読み込む方法は後述します。

さて，それでは，いったいどの機能をカーネルに組み込み，どれを組み込まないようにするべきでしょうか。前述のようにLinuxのカーネルソースには多くの機能が含まれています。また，システムのハードウェア構成や使用環境もさまざまなので，ここですべての項目を説明するのは不可能です。そこで，私がカーネル構築の設定をするとき，どのような基準で判断しているか，簡単にまとめてみました。

ハードウェアのデバイスドライバは使用するものを組み込む。使用する可能性のあるものはモジュール化する。不必要なデバイスドライバを外すことでカーネルのスリム化ができます。

困ったら?でヘルプを読む。説明を読んでもわからないものに関しては，“If unsure, say N.”などの記述がヘルプの中にないか探す。この例では，「確信がもてなければNと答えよ。」とあるので組み込まない。

良くわからないものはデフォルトのままにする。

参考になれば良いのですが…。

また，JFプロジェクト[1]によって，カーネルのアーカイブファイルに含まれるドキュメント（Documentationディレクトリ中のドキュメント）の日本語訳が公開されています[2]。これらのドキュメントも設定の参考にしてください。さらに，?を押したときのヘルプの日本語訳が公開されています[3]。該当のバージョンの翻訳ファイルConfigure.help-x.y.z.ja.gz（x.y.zはカーネルのバージョン）を取得した後，

```
% zcat Configure.help-2.4.19.ja.gz > ~/kernel/linux-2.4.20/Documentation/Configure.help
```

と，付属のConfigure.helpファイルを置き換えることで利用できます。この例では，カーネルのバージョンは2.4.20ですが，該当するConfigure.helpの訳が原稿執筆時点では未公開のため，一番近いバージョン2.4.19のものを使っています。

make configでは，設定を間違えた場合でも前の項目に戻ることはできません。間違えても，とりあえず最後まで進んで設定を終了してください。続いて，make menuconfigを用いて設定の確認及び修正を行います。実行方法は，make configと同様にカーネルのアーカイブファイルを展開したディレクトリで，

```
% make menuconfig
```

です。正しく動作したときは，図２のような画面になります。動作しなかった場合は，コマンドを実行したカレントディレクトリのパスや必要なDebianパッケージがインストールされているかを確認してください。特に，libncurses-devパッケージはインストールされていなくてもmake configは動作しますが，make menuconfigは動きません。

make menuconfigでは，設定項目が階層化されて表示されます。項目間のカーソル移動は，キーボードの上下カーソルキーを用います。また，下位の階層がある場合，項目名の最後に“--->”と表示され，その項目にカーソルを合わせてリターンキーを押すことで下の階層に移動できます。このとき，画面下部に表示される“<Select>”，“<Exit>”，“<Help>”には左右カーソルキーを用いて，“<Select>”にカーソルを合わせておきます。上の階層に戻るには，左右カーソルキーを用いて，画面下部の３つの項目のうち“<Exit>”にカーソルを合わせ，リターンキーを押します。また，最上位階層で“<Exit>”を選択すると，make menuconfigは終了します。終了時には，設定を保存するかの選択が聞かれます。

各設定項目の行頭には，< >もしくは[ ]の括弧が表示されています。< >がモジュール化可能な項目，[ ]ができないものです。項目にカーソルを合わせ，組み込む時はyを，モジュール化するときはm，カーネルから外すときはnを入力します。カーネルに組み込まれる項目には，モジュ

1 http://www.linux.or.jp/JF/
2 http://www.linux.or.jp/JF/JFdocs/kernel-docs-2.4/index.html
3 http://www.linux.or.jp/JF/JFdocs/Configure.help/

ール化される項目にはMが表示されます。また，?でヘルプを表示することができます。

make menuconfigによる設定の確認と修正が済んだらカーネルの設定は終了です。設定内容は，.configという名前のファイルに保存されています。また，Debianの場合，システムにインストールされているカーネルの設定は，/boot/config-x.y.z（x.y.zはカーネルのバージョン）に保存されているので，その設定を引き継いだ設定をすることができます。過去の設定を引き継いだカーネル設定をするには以下のようにします。make oldconfigでは，新たに追加された機能の質問項目のみが表示され，短い時間で設定を行うことができます。

```
% cp /boot/config-2.4.19 ~/kernel/linux-2.4.20/.config
% cd ~/kernel/linux-2.4.20
% make oldconfig
```

Ⅳ．カーネルの構築

設定が終わったら，いよいよカーネルの構築です。古いLinuxディストリビューションでは，カーネルの構築に少し複雑な手順が必要でしたが，Debianでは，make-kpkgコマンドによって自動化されています。なお，make-kpkgを実行するとき，--revisionオプションに日付などを用いたリビジョンナンバを指定してください。例えば，カーネル構築を行う日が，2003年１月１日の場合，以下のようになります。

```
% fakeroot make-kpkg --revision 20030101 kernel_image
```

先頭のfakerootは，Debian固有のおまじないです。忘れるとmake-kpkgの処理の最後に失敗するので，必ずつけるようにしてください。

構築が終了すると，カーネルのソースディレクトリの1つ上の階層のディレクトリにkernel-image-x.y.z_<revision>_i386.deb（x.y.zはカーネルのバージョン，<revision>は上で指定したリビジョンナンバ）という名前でカーネルのDebianパッケージが生成されます。

```
% cd ..
% ls *.deb
kernel-image-2.4.20_20030101_i386.deb
```

Ⅴ．カーネルのインストールと再起動，そして，起動に失敗したら

構築したカーネルのインストールは，rootになりdpkgコマンドを用いて，生成されたカーネルパッケージをインストールするだけです。

```
% su
# dpkg -i kernel-image-2.4.20_20030101_i386.deb
```

なお，前述したようにセキュリティなどの理由で，構築をカーネルをインストールするシステムとは別の計算機上で行った場合は，生成されたカーネルパッケージをターゲットのシステムに転送した後，インストールします。

dpkg -iを実行すると，緊急時やハードディスク以外からブートするときに使用するブートフロッピーを作成するかと聞かれます。必要なら作成してください。また，ブートローダにliloを使っている時は，liloのインストールの実行を聞かれますので，Yesを選択します。

```
Would you like to create a boot floppy now?
Install a boot block using the existing /etc/lilo.conf?
```

また，リビジョンが違っても，過去に同じバージョンのカーネルパッケージがインストールされていると，

```
(…前半略)
Otherwise, I suggest you move /lib/modules/2.4.20 out of the way,
perhaps to /lib/modules/2.4.20.old or something, and then try
re-installing this image.
Do you want to stop now? [Y/n]
```

といった感じの警告が表示されて，インストール処理が中断します。この場合，あらかじめ，/lib/modules/x.y.z（x.y.zはカーネルのバージョン）ディレクトリを他の名前にリネームしてから再度，dpkg -iによるインストールを実行してください。

```
# mv /lib/modules/2.4.20 /lib/modules/2.4.20.old
```

カーネルパッケージのインストールが終了したら，システムを再起動します。

```
# reboot
```

無事に新しいカーネルでシステムは起動したでしょうか?

不幸にも新しいカーネルでの起動に失敗した場合，以前の古いカーネルでシステムを起動して，システムの修復をすることになります。Debianでは，1つ前のカーネルは，/vmlinuz.oldとい

うファイル名で参照可能になっており，ブートローダにliloを標準設定で使用時には，LinuxOLDというラベルで起動することで，この古いカーネルを使用することができます（/etc/lilo.confを参照してください）。liloの詳しい使用法は，オンラインマニュアル[4]などを参照していただくとして，標準的なDebianのシステムでは，とりあえず以下のように起動することで，古いカーネルでのシステムの立ち上げを行うことができます。

1．システムの起動時（BIOSの後），LILO 22.2という表示が画面左上に出たら，すばやくキーボードのシフトボタンを押す。
2．LILO Boot Menuから，LinuxOLDをカーソルで選び，リターンキーを押す。
3．古いカーネルでシステム起動後，再びカーネルの設定を行う。

なお，カーネルパッケージの差し換えをいろいろと行っていると，まれに/vmlinuz.oldという名前で古いカーネルが参照できなくなるときがあります。その場合，手動でカーネルの実体に対してリンクをする必要があります。カーネルの実体は，/bootディレクトリの下にあるvmlinuz-x.y.z（x.y.zはカーネルのバージョン）ファイルです。/vmlinuz.oldとリンクを張るには，

```
# ln -s /boot/vmlinuz-2.4.19 /vmlinuz.old
```

のようにします。なお，手動でリンクをやり直したときには，再起動をする前に，

```
# /sbin/lilo
```

を必ず実行してください。

．モジュールの設定と外部モジュールの構築

カーネル構築時にモジュール化を指定した項目の各モジュールは，/lib/modules/x.y.z（x.y.zはカーネルのバージョン）ディレクトリの下にインストールされます。ディレクトリの下は，さらにカテゴリごとにディレクトリがわけられ，モジュールが.o拡張子付きのファイル名で収められています。また，/lib/modules/x.y.z/modules.depファイルには，それぞれのモジュールのパスと依存関係が記録されています。このファイルなどを参照し，自分が必要とするモジュールのファイル名を確認してください。例えば，Intel PRO/100ネットワークインタフェースカード（EtherExpress Pro/10）のデバイスドライバはeepro100.o，もしくはカーネル2.4.20で追加された新しい実装のデバイスドライバではe100.oになります。

モジュールを手動で読み込むにはmodprobeコマンド，アンロードするにはrmmodコマンドを用います。

4　http://www.linux.or.jp/JM/html/lilo/man8/lilo.8.htm

```
# /sbin/modprobe eepro100
# /sbin/rmmod eepro100
```

また，現在，読み込まれているモジュールの一覧を表示するには，lsmodコマンドを用います。

```
% /sbin/lsmod
```

モジュールを自動的に読み込むには，動的ロードと静的ロードの２種類の方法があります。

静的ロードでは，システムが起動時に自動的に読み込むモジュールを指定します。例えば，SCSIカードのデバイスドライバをモジュール化した場合，起動時にそのモジュールをロードしないとHDDにアクセスできなくなり，起動できなくなることがあります。ネットワークカードのデバイスドライバも起動時に読み込むようにした方が良いでしょう。静的ロードの設定は，modconfコマンドで行います。

```
# modconf
```

modconfを使うことで，メニュー形式でシステム起動時に自動的に読み込むモジュールの指定と解除の設定をすることができます。

一方で，動的ロードでは，モジュールが必要になったときにシステムが自動的にロードをします。設定に関しては，複雑なのでここでは詳細を述べませんが，Debianでは多くの動的ロードの設定がパッケージのインストールのときに自動的に行われるようになっているので便利です。自前で動的ロードの設定をするときは，/etc/modutilsディレクトリ以下の設定ファイルを編集した後，update-modulesコマンドを実行します。

さらにDebianには，カーネルの標準には含まれていない外部デバイスドライバのパッケージが用意されています。PCMCIA（PCカード）ドライバ（パッケージ名：pcmcia-source），高機能サウンドドライバALSA[5]（パッケージ名：alsa-source）などが代表的です。これらドライバもカーネルモジュールとして提供されており，使用するためにはmake-kpkgコマンドを用いた外部モジュールパッケージの構築が必要です。

構築の手順はどれも同じですので，pcmcia-sourceを例に説明します。まず，apt-getを用いてパッケージを取得します。

```
# apt-get install pcmcia-source
```

アーカイブファイルが/usr/srcディレクトリ以下に生成されます。このアーカイブをカーネル

5　http://www.alsa-project.org/

構築のときに用いた作業用ディレクトリ（前述の例では，~/kernel）に展開します。カーネルの構築時と同様に一般ユーザ権限で作業するように心がけてください。

```
% cd ~/kernel
% tar xvzf /usr/src/pcmcia-cs.tar.gz
```

~/kernel/modules/以下にソースが展開されます。ALSAなど他に必要な外部モジュールがある場合は，同様に~/kernel/modules/にすべて展開してください。

続いて，この原稿の前半でカーネルを構築したディレクトリ（~/kernel/linux-2.4.20）にカレントディレクトリを移動します。

```
% cd ~/kernel/linux-2.4.20
```

そして，環境変数MODULE_LOCにモジュールのソースを展開したディレクトリのパスを指定して，make-kpkgコマンドを実行します。環境変数の指定方法は，お使いのシェルによって異なります。例えば，bashなどBシェル系では，

```
% MODULE_LOC=’../modules’ fakeroot make-kpkg modules_image
```

Cシェル系では，

```
% setenv MODULE_LOC ‘../modules’
% fakeroot make-kpkg modules_image
```

となります。

しばらくすると処理が終了して，１つ上の階層のディレクトリにモジュールごとにDebianパッケージが生成されます。最後にパッケージをdpkg -iを用いてインストールします。

```
% cd ..
% ls *.deb
kernel-image-2.4.20_20030101_i386.deb
pcmcia-modules-2.4.20_3.1.33-6+20030101_i386.deb
% su
# dpkg -i pcmcia-modules-2.4.20_3.1.33-6+20030101_i386.deb
```

ここでパッケージに付与されるリビジョンナンバは，さきほどカーネルを構築したときに

make-kpkgコマンドの--revisionオプションで指定したものです。設定内容を変更したときなどは，リビジョンナンバを変えて，カーネル及び外部モジュールを再構築することになります。その際には，一度，

```
% fakeroot make-kpkg clean
```

を実行して，中間生成ファイルなどをすべて消去した状態から，再びカーネルの構築を行ってください。

．ユーザ権限の切り替え：su と sudo

少しおまけです。今回の説明では，カーネルの構築をrootではない一般ユーザで行いました。そして，dpkg -iでのパッケージのインストールなど，root権限が必須のときのみroot権限を使っています。具体的には，コマンドプロンプトが，%で書かれている部分が一般ユーザ権限での操作であり，#のものがroot権限を必要とする操作です。

一般ユーザからrootに権限を切り替えるときに使用するコマンドがsuです。普段は，一般ユーザで操作して，本当に必要なときのみsuを用いてrootに一時的にユーザ権限を変更します。そして，root権限の必要がなくなったら，exitコマンドでrootをぬけて一般ユーザに戻ります。このように必要なときのみrootになるという心がけは，みなさんのセキュリティに対する意識を高めるうえで重要なことですから，必ず実践するようにしてください。くれぐれもだらだらとrootで作業をしないように。

また，suに代わる便利なツールとしてsudoというコマンドがあるので，そちらを使うようにすると良いでしょう。sudoのインストールはいつものようにapt-getを使います。

```
# apt-get install sudo
```

続いて，visudoコマンドを用いて設定を行います。

```
# visudo
```

エディタが立ち上がりますので，最後の行に

```
<user>    ALL=(ALL) ALL
```

と追加します。<user>の部分には，自分のアカウントのユーザ名を記述します。例えば，私のユーザ名は，nisimuraですから，以下のようになります。

```
nisimura    ALL=(ALL) ALL
```

変更内容を保存して，エディタを終了したら設定は終りです。それではsudoを使ってみましょう。使い方は簡単で，root権限を必要とするコマンドの頭にsudoを追加するだけです。例えば，kernel-image-2.4.20_20030101_i386.debをdpkg -iでインストールする操作は以下のようになります。

```
% sudo dpkg -i kernel-image-2.4.20_20030101_i386.deb
```

実行すると，パスワードの入力が求められます。ここでsuとは違いrootのパスワードではなく，自分自身の一般ユーザのログインパスワードを入力します。正しいパスワードが入力された後，root権限でコマンドは実行され，終了するとまた一般ユーザに戻ります。つまり，visudoによって許可されたユーザはrootのパスワードを知ることなく，root権限を行使することができます。一度正しいパスワードを入力すると，タイマーによりしばらくの間はパスワードの入力が省略されるので利便性が損なわれることはありません。このように明示的にコマンドの頭にsudoを付けることにより，必要なときだけroot権限を使うように心がけましょう。

また，今回のvisudoの設定では，ユーザに対してすべての操作をroot権限で行えるように許可しましたが，特定のコマンドのみをroot権限で実行できるようにしたり，柔軟な制限を設定することが可能です。詳しくは，sudo，sudoers，visudoのオンラインマニュアル[6] [7] [8]を参照してください。

さらにもう一つDebianに関するマメ知識です。visudoを使ったときに立ち上がったエディタは，システム全体で標準のエディタとして設定されているものです。この標準エディタの設定は，

```
% sudo update-alternatives --config editor
```

で変更することができます。ちなみに，私はvi互換のnviというエディタを使っているので，

```
% sudo apt-get install nvi
```

でインストールしてから使っています。sudoを使うことによりapt-getの実行も一般ユーザで行っていることをわかっていただけたでしょうか?

6 http://www.linux.or.jp/JM/html/sudo/man8/sudo.8.html
7 http://www.linux.or.jp/JM/html/sudo/man5/sudoers.5.html
8 http://www.linux.or.jp/JM/html/sudo/man8/visudo.8.html

．今日のおさらい

最後に今日のおさらいです。カーネルのアーカイブファイルの取得から設定，構築，インストールまでの手順の流れをまとめておきますので，参考にしてください。

```
% mkdir ~/kernel
% cd ~/kernel
% wget http://www.ring.gr.jp/pub/linux/kernel.org/kernel/v2.4/linux-2.4.20.tar.bz2
% tar xvjf linux-2.4.20.tar.bz2
% cd linux-2.4.20
% make config
% make menuconfig
% fakeroot make-kpkg --revision 20030101 kernel_image modules_image
% cd ..
% sudo dpkg -i kernel-image-2.4.20_20030101_i386.deb
% sudo reboot
```

．おわりに

今回は，本当はiptablesの話をするつもりでしたが，その前段階のカーネルの再構築の話だけで時間になってしまいました。次回こそは，iptablesの話をしたいと思います。本連載では，できるだけ丁寧な説明を心がけているのですが，説明が無駄に長くなっている部分があると思います。なかなか話がすすまなくて書いている本人もイライラしています(^^;;。連載はもう少し続けるつもりでいますので，みなさまのご意見やご要望などありましたら，末尾のメールアドレスまでお寄せください。

（にしむら りゅういち：奈良先端科学技術大学院大学情報科学研究科）
（nisimura@linux.or.jp）