# カーネルビルド入門

67 回生 hiromu1996

## Ⅰ：はじめに

こんにちは、@hiromu1996 です。今回は、Linux の中核部分である Linux カーネルを自分でビルドする方法を説明していきます。

そもそも、Linux とは 1991 年に Linus Torvalds によって開発された OS です。Windows のように PC にいれて使えるだけじゃなく、最近流行しているスマートフォンなんかにも搭載されています。しかも、オープンソースソフトウェアなので、誰でもソースコード（＝設計図）を読むことが出来て、無料で使うことができます。さらに、世界中の人と共に開発に参加して、だれでも改良していくことができます。かく言う私も開発者の一人です。

それでは説明を始めましょう。（以下、Linux 環境での作業を想定しています。）

## Ⅱ：ソースコードを入手する

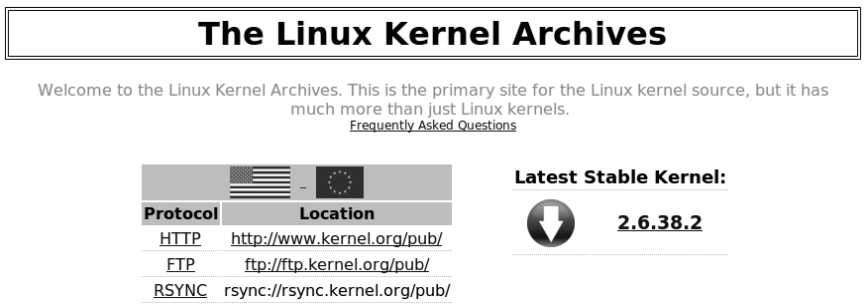
カーネルのビルドをするには設計図となるソースコードを入手せねばなりません。これには大きく分けて 2 つの方法があります。

1. tar ball をダウンロードする

   http://www.kernel.org/のトップページ（下の画像）の「Latest Stable Kernel」のところからダウンロードできます。現在の stable 版（＝安定版）のバージョンは 2.6.38.2 なので、そこをクリックすると linux-2.6.38.2.tar.bz2 のダウンロードが始まります。

   ダウンロードが完了すれば以下のようにして解凍します。なお、ソースコードは/usr/src/に展開するのが通例となっています。

   ```
   $ tar -jxvf <ダウンロードしたファイル名> -C /usr/src/
   ```



2. git を使ってダウンロードする

git というバージョン管理システムを使ってリポジトリ(＝ソースコードを情報や記録を含めて管理している場所)ごとコピーします。

はじめに git のシステムをインストールする必要があります。

パッケージ管理に apt を使っているなら

```
# apt-get install git
```

yum を使っているなら

```
# yum install git
```

portage を使っているなら

```
# emerge git
```

次に/usr/src 以下にリポジトリをクローンします。

現在の stable 版のリポジトリは linux-2.6.38.y.git ですが、最新のリポジトリの場所は http://www.kernel.org/から調べることができます。

```
# cd /usr/src
# git clone git://git.kernel.org/pub/scm/linux/
  kernel/git/stable/linux-2.6.38.y.git
```

## Ⅲ：ソースコードの構成

ソースコードの入手ができたら、中身をすこし見てみましょう。Linux カーネルは基本的に C 言語で書かれています。中に arch、block、crypto……といった 20 個ほどのディレクトリがありますが、それぞれについて軽く説明していきます。

Documentation - カーネルに関するドキュメント類。

arch - アーキテクチャに依存するコード。x86 なら arch/x86、mips なら arch/mips にあるファイルが参照される。

block - ブロックデバイス(＝ハードディスクは CDROM など)に関するコード・

crypto - 暗号化や署名、圧縮に関するコード。

drivers - いろいろなデバイスのドライバが入っている。例えば、drivers/net 以下にはネットワークデバイスのドライバがある。

firmware - 古いドライバがバイナリのままある。

fs - ファイルシステムに関するコード。ext2、ext3、btrfs などいろいろなファイルシステムにアクセスするのに必要。

include - 他のコードから参照される部分。

init - カーネルの開始や初期化に関するコード。

ipc - プロセス間通信に関するコード。

kernel - カーネルのコアとなる機能に関するコード。スケジューラーやタイマー、割り込み、CPU 制御など非常に重要な部分である。

lib - 他のコードで使用されるライブラリがある。

mm - メモリ管理に関するコード。mmap や OOM Killer、swap なども実装されて いる。

net - IPv4 や IPv6、無線ネットワークなどのネットワークに関するコードがある。

samples - カーネルのサンプルコードがある。

scripts - カーネルのビルドや開発に使用されるスクリプトがある。

security - SELinux や TOMOYO Linux といった Linux Security Module など。

sound - サウンドに関するコード。サウンドチップのドライバもある。

tools - パフォーマンスカウンターなどのツールがある。

usr - initramfs(＝起動時に使用する)に関するコード。

virt - KVM(kernel virtual machine)に関するコード。

このように 35865 個ものファイルが分類されています。C 言語をかける方はすこしソースコードを見てみると面白いかもしれません。

## Ⅳ：カーネルの設定

カーネルをビルドするまえに、カーネルを設定する必要があります。たくさん用意されている設定方法の中からいくつかを紹介しましょう。

```
$ make config
```

一つ一つの項目についてコンソール上で設定していく。余り使わない

```
$ make menuconfig
```

コンソール上でメニュー形式で設定する。

```
$ make gconfig
```

gtk というライブラリをつかって GUI で設定できる。

```
$ make oldconfig
```

古い設定に基づいて、一つずつ設定していく。バージョンアップの時などに使う。

```
$ make defconfig
```

デフォルトの設定が生成される

```
$ make localmodconfig
```

現在使用している設定をもとに変更を加える。


設定方法の例

順に make config　make menuconfig

　　make gconfig

その他にもいろいろな設定方法があるのですが、基本的には上記のものを使います。それでは、カーネルの設定をしていきましょう。私は、"make localmodconfig" で設定を生成したあとに"make menuconfig" でいらないものを削っていくという形でいつも設定しています。"make localmodconfig" は現在使用しているモジュールも自動で有効化するので、設定に失敗して起動しなくなることが少ないです。また"make menuconfig" でいらないものを削ると、カーネルを軽くすることができます。それでは"make menuconfig" での設定項目について説明していきます。

* General setup - カーネルの一般的な設定。Swap や初期 RAM ディスクの無効化ができるが、重要な設定ばかりなので削らない方がよい。
* Enable loadable module support - カーネルモジュールの設定。ドライバなどで使用することが多いので有効にしておくべき。

* Enable the block layer - ブロックデバイスと IO スケジューラーの設定。
* Processor type and features - プロセッサの種類や機能に関する設定。”Processor family” を利用している CPU に設定しておくとよい。
* Power management and ACPI options - ACPI などの電源管理に関する設定。ハイバーネートするには”Hibernation (aka 'suspend to disk')” を選択。
* Bus options (PCI etc.) - PCI や PCI Express の設定。
* Executable file formats / Emulations - 実行ファイル形式の設定。ELF や IA64 形式を動かすには設定する必要がある。
* Networking options - ネットワークの機能の設定。サーバーなどで iptables や IPv6 を使用するときは”Networking options “を細かく設定すると良い。また、Bluetooth や WiMAX を使うときはそれぞれ有効化する必要がある。
* Device Drivers - デバイスドライバの設定。それぞれのハードウェアに合わせて設定を削ると大変軽くなる。しかし、ディスクのドライバなどを間違えて削ってしまうと起動しなくなるなるので、デバイスの種類を考えて削るとよい。
* Firmware Drivers - BIOS や EFI の機能に関する設定。
* File systems - ファイルシステムのサポートに関する設定。ディスクで使用しているファイルシステムは有効化する必要がある。
* Kernel hacking - カーネルデバッグに使う機能を設定する。
* Security options - セキュリティに関する設定。SELinux や TOMOYO Linux などの LSM も設定できる。
* Cryptographic API - カーネルでの暗号化機能に関する設定。
* Virtualization - 仮想化に関する設定。KVM を使うには有効化する必要がある。
* Library routines - ライブラリモジュールに関する設定。

設定が終わったら保存して終了しましょう。「.config」というファイルができているはずです。

## ∇：カーネルビルドとインストール

それでは、ビルドを始めましょう。

```
$ make
```

しばらくの間は「CC ＊＊＊＊＊.o」みたいな文字が流れているので、終わるまで待ちましょう。終わると次はカーネルをインストールします。

まず、カーネルモジュールをインストールします。(root 権限が必要です。)

```
# make modules_install
```

次に、カーネル本体をインストールします。

```
# make install
```

さらに、debian 系では initramfs(＝起動時に使用するファイルシステム)を更新します。

```
# update-initramfs -u -k <カーネルバージョン>
```

そしてブートローダーも更新する必要があります。grub を使っているなら、/boot/grub/grub.conf が更新されているのを確認します。また grub2 を使っているなら次のようにして更新します。

```
# update-grub2
```

これでインストールは完了です!さあ、起動することを祈りつつ再起動してみましょう!

## 番外編：カーネル開発

ページがすこし残っているのでカーネル開発についての話を書こうと思います。カーネル開発は LKML(Linux Kernel Mailing List)というメーリングリストが中心となって行われています。LKML では一日約 400 通、年間 150000 通あまりのメールがやり取りされているのです。もちろんこの中には Linux を作った Linus Torvalds のメールもあります。LKML で新機能の検討がされたり、リリースの日程を決めたりということが行われているのです。

それでは、開発に参加するにはどうすればいいのでしょうか。例えば、Linux のソースコードの中にバグを見つけたとします。そして、あなたはそれを改善するコードを書きました。それからどうすればいいのでしょうか。答えは、パッチ形式にして LKML に送るのです。しかし、いろいろな注意点があります。パッチをどうやって作って、どうやって送ればいいのか。簡単に説明しましょう。

まず、git ツリーからパッチを作ります。

```
$ git format-patch <リビジョン>
```

git のコマンドでパッチを作ることができます。しかし、パッチの形式にも注意しなくてはいけません。Unified diff 形式でないと適用するのが面倒なので相手も困ります。また、Documentation/CodingStyle に書いてあるコーディングルールを守るのも非常に重要です。そのため、送る前にパッチをチェックしておきましょう。

```
$ scripts/checkpatch.pl <パッチファイル>
```

scripts/checkpatch.pl を使用するとパッチのインデントや括弧の場所、形式などをチェックできます。また、パッチが大きい場合は複数のファイルに分けるべきです。あまりに大きいと読みにくく、なかなか他の人に見てもらえません。それに、一つの変更ごとに絶対パッチを分けましょう。複数の変更を一つのパッチに入れてしまうと、あとから変更点が分かりにくくなります。

パッチができたらLKML にメールを送りましょう。ここでも、いくつかの注意があります。

・適用されるバージョンを明示すること

・件名の先頭に[PATCH]という文字を含めること

・Developer's Certificate of Origin に乗っとって署名をすること

・パッチは添付ファイルとして送るのではなく、インラインで送ること

・HTML メールではなくPlain Text で送ること

・一つのパッチに対し、一通ずつメールを送ること

上の注意点を守ってメールが書けたら、宛先を選びましょう。To にはLKML のメールアドレス、つまり linux-kernel@vger.kernel.org を、CC にはサブシステムの ML やメンテナー（＝そのシステムの管理を担っている人）、最近変更をした人などを選ぶといいでしょう。sclipts/get_maintainer.pl を使うとメンテナーのメールアドレスがわかるので、これも利用するといいでしょう。

それではメールを送りましょう！返事があるまで、急かさずに忍耐強く待ちましょう。何らかの反応があるはずです。たとえ、それが反対意見だったとしても、原因をさがし修正してもう一度送ってみましょう。送ったコードがLinux にとってすこしでもいいものであれば、絶対メインのリポジトリに載せてもらえます。その瞬間から、あなたはれっきとしたLinux のコントリビューターです！さあ、カーネル開発を始めてみませんか？

## Ⅵ：さいごに

カーネル開発は大変面白いです。普段使っているLinux がどのような仕組みで動いているのかも理解できます。そして、コントリビューターになると、世界中で自分の書いたコードが使われているということになります。つまり、クラスメイトがもっているスマートフォンでも、Wikipedia のサーバーでも自分の書いたコードが入っているのです！わくわくしませんか！

僕と契約して、Linux カーネルのコントリビューターになってよ！