Linuxカーネルへのシステムコール実装手順書

2014/4/21

藤田将輝

1 はじめに

本手順書ではLinuxカーネルへのシステムコール実装の手順を示す. 以下に章構成を示す.

第1章 はじめに
第2章 実装環境
第3章 カーネルのソースコードの取得
第4章 システムコールの実装手順
第5章 注意すべき点
第6章 おわりに

2 実装環境

実装環境を表1に示す.

表1 実装環境

項目名	環境
OS	Fedora14
カーネル	Linuxカーネル 3.0.8 64bit
CPU	Intel(R) Core(TM) Core i7-870 @ 2.93GHz
メモリ	2GB

3 カーネルのソースコードの取得

Gitを用いて, カーネルのソースコードを取得する. Gitはバージョン管理ツールであり,git clone [リポジトリのURL] を実行すると指定したリポジトリを取得する. git checkout -b [ブランチ名] [タグ名] を実行すると指定したブランチ名のブランチを作成し, 切り替える. また, タグを指定すると, 特定の状態からブランチを作成できる. ブランチとはGitにおける開発ラインのことである. 以下に実行例を示す.

```
$ git clone git://git.kernel.org/pub/scm/linux/kernel/git/ \
```

```
  stable/linux-stable.git Linux-3.0.8
$ cd Linux-3.0.8/
$ git checkout -b linux3.0.8 v3.0.8
```

実行例では Linux-3.0.8 にリポジトリを取得し, linux3.0.8 というブランチを作成し, 切り替えている.

4 システムコールの実装手順

4.1 カレントディレクトリ

カレントディレクトリは/home/fujita/git/Linux-3.0.8 とする.

4.2 システムコールの追加

システムコールの追加手順について以下で説明する. 書き加えた行には + を付与している.

(1) システムコール本体の作成
arch/x86/kernel 以下にシステムコール本体を作成する. 例として test.c というファイルを作成し,sys_test という名前のシステムコールを作成する. これは, 整数を引数にとり, この整数の 2 倍の値を返すものである. 以下にソースコードを示す.

```
1  int sys_test(int arg1)
2  {
3    return arg1 + arg1;
4  }
```

(2) ヘッダ・ファイルの編集
unistd_64.h と syscalls.h を編集する.

（A）unistd_64.h の編集
arch/x86/include/asm/unistd_64.h に追加するシステムコールの名前とシステムコール番号をつけ, 書き加える. 例ではシステムコール番号として,__NR_test を 309 と定義し,sys_test を追加している. システムコール番号は既存のシステムコール番号と重複しないようにする. 以下に例を示す.

```
 682     #define __NR_setns                              308
 683     __SYSCALL(__NR_setns, sys_setns)
+684     #define __NR_test                               309
+685     __SYSCALL(__NR_test, sys_test)
```

（B）syscalls.h の編集
arch/x86/include/asm/syscalls.h にシステムコールのプロトタイプ宣言を追加する. 以下に例を示す.

```
 43     asmlinkage int sys_get_thread_area(struct user_desc __user *);
+44     extern int sys_test(int arg1);
```

(3) Makefile の編集

arch/x86/kernel/Makefile にシステムコールのオブジェクトファイル名を追加する．以下に例を示す．

```
 117    obj-$(CONFIG_OF)    += devicetree.o
+118    obj-y               += test.o
```

4.3 カーネルの再構築

カーネルの再構築の手順を以下に示す．

(1) カーネルの設定ファイルの作成

カーネルの設定ファイル.config を作成する．

（A）ncurses-devel のインストール

make menuconfig を実行するために必要な ncurses-devel をインストールする．ncurses-devel はカーネルの設定画面を表示するために必要なパッケージである．以下に実行例を示す．

```
$ sudo yum install ncurses-devel
```

（B）make menuconfig の実行

以下のコマンドでカーネルを設定する．

```
$ make menuconfig
```

コマンドを実行するとカーネルの設定画面に切り替わる．Fedora14 では標準ファイルシステムが ext4 であり, ext4 に対応するようにカーネルを設定する．File system 項目で以下のように設定する．組み込む項目には [*] を付ける．なお, 組み込むか否かはスペースキーの押下で変更する．

```
<*>The Extend 4 (ext4) filesystem
[*]  Use ext4 for ext2/ext3
[*]  Ext4 extended attributes
[*]    Ext4 POSIX Access Contorol Lists
[*]    Ext4 Security Labels
```

設定後,Esc キーを入力しホーム画面に戻る．Exit を選択し, 終了する．すべての設定が終了するとカレントディレクトリ以下に.config が作成される．

(2) bzImage の作成

bzImage は Linux カーネルである vmlinux を圧縮した形式のファイルである．以下に bzImage を作成する実行例を示す．なお,j8 はコンパイルを並列処理し, コンパイル時間を短縮するオプションである．

```
$ make bzImage -j8
```

この操作を行うことで,bzImage が arch/x86/boot 以下に作成される．さらにカーネルが使用するシンボルテーブルである System.map がカレントディレクトリ以下に作成される．

(3) カーネルモジュールのコンパイル, インストール
カーネルモジュールはカーネルの機能を拡張するもので, 必要に応じてカーネルにロード, アンロードできる．例ではインストールの実行を終えると DEPMOD 3.0.8+ と表示される．この 3.0.8+ はモジュールがインストールされたディレクトリ名である．これは/lib/modules 以下に作成される．以下にカーネルモジュールをコンパイルし, インストールするコマンドを示す．

```
$ make modules
$ sudo make modules_install
```

実行結果例を以下に示す．

```
  INSTALL sound/usb/snd-usbmidi-lib.ko
  INSTALL sound/usb/usx2y/snd-usb-us122l.ko
  INSTALL sound/usb/usx2y/snd-usb-usx2y.ko
  DEPMOD  3.0.8+
```

(4) カーネルのインストール
(1) で作成した bzImage と System.map に名前を付けて/boot 以下にコピーする．例では作成した bzImage と System.map にそれぞれ vmlinuz-3.0.8-fujita,System.map-3.0.8-fujita という名前を付ける．カーネルをインストールする実行例を以下に示す．

```
$ sudo cp arch/x86/boot/bzImzge /boot/vmlinuz-3.0.8-fujita
$ sudo cp System.map /boot/System.map-3.0.8-fujita
```

(5) 初期 RAM ディスクの生成
初期 RAM ディスクとは実際のルートファイルシステムが使用できるようになる前にマウントされる初期ルートファイルシステムである．以下に初期 RAM ディスクを作成するコマンドを示す．

```
$ sudo mkinitrd /boot/initrd-3.0.8-fujita.img 3.0.8+
```

この 3.0.8+ の部分は make moduls_install を実行したときに表示される DEPMOD に記されているディレクトリを指定する．3.0.8+ の中にはカーネルモジュールがインストールされており, これをもとに初期 RAM ディスクを作成する．例では初期 RAM ディスク initrd-3.0.8-fujita.img を作成している．

(6) ブートローダの設定
再構築したカーネルを起動するために,GRUB の設定ファイルにエントリを追加する．GRUB とはブートローダであり, ブート処理を行うものである．/boot/grub/menu.lst を編集することで,

設定を変更できる. 編集にはルート権限が必要である. 以下に追加するエントリの例を示す.

```
1    title Linux-3.0.8-fujita
2            root (hd0,0)
3            kernel /vmlinuz-3.0.8-fujita                                    \
4            ro root=UUID=56a552db-4819-4fa4-8b8e-5f76afb7edd7               \
5            rd_NO_LUKS rd_NO_LVM rd_NO_MD rd_NO_DM LANG=en_US.UTF-8         \
6            SYSFONT=latarcyrheb-sun16 KEYTABLE=jp106 rhgb quiet
7            initrd /initrd-3.0.8-fujita.img
```

1 行目の title はカーネルの選択画面で表示されるエントリ名であり, 任意のエントリ名を入力する. 2 行目の root は/boot のパーティションを指定する. 書式は (ハードディスクの番号, パーティション番号) となっている. パーティションとはハードディスクの分割された個々の領域のことである. 例では 1 番目のハードディスクの 1 番目のパーティションを指定している. 3 行目の kernel は起動するカーネルを指定する. 今回の例では (3) で名前を付けた vmlinuz-3.0.8-fujita を指定している. 4,5,6 行目はブートオプションである. 4 行目の ro root にはルートのパーティションの UUID を指定する. UUID は一意な識別子である. 5,6 行目は既存のエントリと同様に設定する. 7 行目は起動時に使用する初期 RAM ディスクを指定する. 例では (4) で作成した initrd-3.0.8-fujita.img を指定している.

(7) 再起動

すべての設定を終えた後, 以下のコマンドで計算機を再起動する.

```
$ sudo reboot
```

4.4 テスト

実装したシステムコールを使ったプログラムを作成し, 実行することでシステムコールをテストする. 今回実装したシステムコールは整数を引数にとり, その整数の 2 倍の値を返す. よって,4 を引数にした場合,8 が出力されればシステムコールの実装の成功となる. システムコールの呼び出しには syscall 関数を用いる. このとき,syscall 関数に 1 つ目の引数として, 実装したシステムコールのシステムコール番号, 2 つ目の引数として, 実装したシステムコールの引数を与える. テストプログラムを以下に示す.

```
1  #include<stdio.h>
2  #include<sys/syscall.h>
3
4  main()
5  {
6          printf("%d\n",syscall(309,4));
7  }
```

5 注意すべき点

カーネルの再構築時に注意すべき点は bzImage や初期 RAM ディスクをインストールする際に生成されるディレクトリやファイルを上書きせずに, 名前を変えて別のカーネルとして保存することである. これにより, 既存の環境を破壊することを防ぐ.

6 おわりに

本手順書では Linux カーネルへのシステムコールの実装手順を示した.