システムコール追加の手順書

2018/4/19

藤原 裕貴

1　はじめに

本手順書では，システムコール追加の手順を説明する．例として，カーネルのメッセージバッファに任意の文字列を出力するシステムコールを追加する．本手順書は，コンソールの基本的な操作を習得している者を読者として想定している．システムコールを追加するためのカーネルの取得から説明を行う．第 8 章には付録として追加したシステムコール本体のソースコードを添付する．以下に，本手順書の章立てを示す．

第 1 章　はじめに
第 2 章　実装環境
第 3 章　環境設定
第 4 章　Linux カーネルの取得
第 5 章　システムコールの追加
第 6 章　テスト
第 7 章　おわりに
第 8 章　付録

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	Debian7.10
カーネル	Linux カーネル 3.15.0
CPU	Intel Core i7-4770 4 コア
メモリ	2GB

3 環境設定

3.1 sudo 権限の付与

ユーザに sudo 権限を与える．以下のコマンドを実行する．

```
$ su
# visudo
```

エディタが起動する．その後，"root ALL=(ALL) ALL"の次の行に"fujiwara-yu ALL=(ALL) ALL"を追加する．これは，"**ユーザ名 ホスト名=(権限) コマンド**"で指定される．なお，fujiwara-yu はユーザ名であり，自身のユーザ名に変更して記述する．

3.2 パッケージのインストール

本手順において必要となるパッケージのインストール方法を示す．以下は git を例とした場合である．コマンドを実行することで，パッケージのインストールが行える．

```
$ sudo apt-get install git
```

システムコールを追加するために必要となるパッケージを以下に示す．

(1) git
(2) gcc
(3) bc
(4) make

git は第 4 章の Linux カーネルの取得で用い，gcc，bc，および make は第 5 章の第 5.2 節のカーネルの再構築で用いる．

4 Linux カーネルの取得

4.1 Linux のソースコードの取得

Linux のソースコードは Git で管理されている．Git とは，オープンソースのバージョン管理システムである．下記の Git リポジトリをクローンし，Linux のソースコードを取得する．リポジトリとは，ファイルやディレクトリを保存する場所のことである．クローンとは既存の Git リポジトリを複製することである．

```
git://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable.git
```

本手順書では，/home/fujiwara-yu/git 以下でソースコードを管理する．/home/fujiwara-yu で以下のコマンドを実行する．

```
$ mkdir git
$ cd git
$ git clone \
  git://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable.git
```

まず，mkdir コマンドにより/home/fujiwara-yu 以下に git ディレクトリが作成される．次に，cd コマンドにより，git ディレクトリに移動する．そして，そのディレクトリで，git clone コマンドを実行し，リポジトリの複製を行う．すると，/home/fujiwara-yu/git 以下に linux-stable ディレクトリが作成される．/home/fujiwara-yu/git/linux-stable 以下に Linux のソースコードが格納されている．

4.2 ブランチの切り替え

Linux のソースコードのバージョンを切り替えるため，ブランチの作成と切り替えを行う．ブランチとは，ソースコードの編集履歴を管理するための分岐である．ブランチを切り替えることで，ブランチごとに異なる編集履歴を管理できる．/home/fujiwara-yu/git/linux-stable で以下のコマンドを実行する．

```
$ git checkout -b 3.15 v3.15
```

実行後，v3.15 というタグが示すコミットからブランチ 3.15 が作成され，カレントブランチがブランチ 3.15 に切り替わる．コミットとは，ある時点における開発の状態を記録したものである．タグとは，コミットを識別するためにつける印である．

5 システムコール追加の手順

5.1 ソースコードの編集

システムコールを追加するために，以下の手順でソースコードの編集を行う．本手順書では，既存のファイルを編集する際，追加した行の先頭に + を付け，削除した行の先頭には − を付けて表す．行の先頭の数字はファイルの行番号である．

(1) システムコール本体の追加

/home/fujiwara-yu/git/linux-stable/kernel 以下にシステムコールのソースファイル mysyscall.c を作成する．以下に追加するシステムコールの概要を示す．

【形式】 asmlinkage void sys_mysyscall(char *str)

【引数】 char *str：任意の文字列

【戻り値】 無し

【機能】　カーネルのメッセージバッファに任意の文字列を出力する．

(2) システムコールのプロトタイプ宣言

/home/fujiwara-yu/git/linux-stable/include/linux/syscalls.h を以下のように編集し，作成したシステムコールのプロトタイプ宣言を行う．

```
  866 asmlinkage long sys_kcmp(pid_t pid1, pid_t pid2, int type,
  867                          unsigned long idx1, unsigned long idx2);
  868 asmlinkage long sys_finit_module(int fd, \
                                  const char __user *uargs, int flags);
+ 869 asmlinkage void sys_mysyscall(char *str);
```

(3) システムコール番号の定義

/home/fujiwara-yu/git/linux-stable/arch/x86/syscalls/syscall_64.tbl を以下のように編集し，作成したシステムコールのシステムコール番号を定義する．システムコール番号は，システムコールとそれぞれ対応しており，この番号を syscall 命令の引数にすることでシステムコールを呼び出せる．そのため，システムコール番号は重複して定義できない．

```
  324 315    common  sched_getattr       sys_sched_getattr
  325 316    common  renameat2           sys_renameat2
+ 326 317    common  mysyscall           sys_mysyscall
```

(4) Makefile の編集

システムコール本体のソースファイルである/home/fujiwara-yu/git/linux-stable/kernel/mysyscall.c のコンパイルを行うために，/home/fujiwara-yu/git/linux-stable/kernel/Makefile を以下のように編集する．

```
   5 obj-y     = fork.o exec_domain.o panic.o \
   6               cpu.o exit.o itimer.o time.o softirq.o resource.o \
   7               sysctl.o sysctl_binary.o capability.o \
                   ptrace.o timer.o user.o \
   8               signal.o sys.o kmod.o workqueue.o pid.o task_work.o \
   9               extable.o params.o posix-timers.o \
  10               kthread.o sys_ni.o posix-cpu-timers.o \
  11               hrtimer.o nsproxy.o \
  12               notifier.o ksysfs.o cred.o reboot.o \
- 13               async.o range.o groups.o smpboot.o
+ 13               async.o range.o groups.o smpboot.o mysyscall.o
```

5.2 カーネルの再構築

カーネルの再構築を行う．/home/fujiwara-yu/git/linux-stable で以下の手順に従ってコマンドを実行する．

(1) .config ファイルの作成
カーネルの設定を記述した.config ファイルを作成する．以下のコマンドを実行し，x86_64_defconfig ファイルを基に，カーネルの設定を行う．x86_64_defconfig ファイルにはデフォルトの設定が記述されている．

```
$ make defconfig
```

実行後，/home/fujiwara-yu/linux-stable 以下に.config ファイルが作成される．

(2) カーネルのコンパイル
以下のコマンドを実行し，カーネルをコンパイルする．

```
$ make bzImage -j8
```

上記コマンドの「-j」オプションは，同時に実行できるジョブ数を指定する．ジョブ数を不用意に増やすとメモリ不足により，実行速度が低下する場合がある．今回は，例としてコア数の 2 倍の 8 としている．コマンド実行後，/home/fujiwara-yu/git/linux-stable/arch/x86/boot 以下に bzImage という名前の圧縮カーネルイメージが作成される．カーネルイメージとは実行可能形式のカーネルを含むファイルである．同時に/home/fujiwara-yu/git/linux-stable 以下にすべてのカーネルシンボルのアドレスを記述した System.map が作成される．カーネルシンボルとは，カーネルのプログラムが格納されたメモリアドレスと対応付けられた文字列である．

(3) カーネルのインストール
コンパイルしたカーネルをインストールする．以下のコマンドを実行する．

```
$ sudo cp /home/fujiwara-yu/git/linux-stable/arch/x86/boot/bzImage \
          /boot/vmlinuz-3.15.0-linux
$ sudo cp /home/fujiwara-yu/git/linux-stable/System.map \
          /boot/System.map-3.15.0-linux
```

実行後，bzImage と System.map が/boot 以下にそれぞれ vmlinuz-3.15.0-linux と System.map-3.15.0-linux という名前でコピーされる．

(4) カーネルモジュールのコンパイル
以下のコマンドを実行し，カーネルモジュールをコンパイルする．カーネルモジュールとは，カーネルの機能を拡張するためのバイナリファイルである．

```
$ make modules
```

(5) カーネルモジュールのインストール
コンパイルしたカーネルモジュールをインストールする．以下のコマンドを実行する．

```
$ sudo make modules_install
```

実行後，出力結果の最後の行は以下のように表示される．

```
DEPMOD 3.15.0
```

上記の DEPMOD 以下の 3.15.0 は，カーネルモジュールをインストールしたディレクトリ名である．このディレクトリ名は手順 (6) で指定するため，控えておく．

(6) 初期 RAM ディスクの作成

初期 RAM ディスクを作成する．以下のコマンドを実行する．

```
$ sudo update-initramfs -c -k 3.15.0
```

コマンドの引数として手順 (5) で控えたディレクトリ名を与える．実行後，/boot 以下に初期 RAM ディスク initrd.image-3.15.0 が作成される．

(7) ブートローダの設定

システムコールを追加したカーネルをブートローダから起動可能にするため，ブートローダを設定する．ブートローダの設定ファイルは/boot/grub/grub.cfg である．カーネルのエントリを追加するには設定ファイルを編集する必要がある．なお，本環境で使用されているブートローダは GRUB2 である．GRUB2 でカーネルの選択画面にエントリを追加する際，設定ファイルを直接編集しない． /etc/grub.d 以下にエントリ追加用のスクリプトを作成し，そのスクリプトを実行することでエントリを追加する．GRUB2 のブートローダを設定する手順を以下に示す．

（A）エントリ追加用のスクリプトの作成

カーネルのエントリを追加するため，エントリ追加用のスクリプトを作成する．本手順書では，既存のファイル名から作成するスクリプトのファイル名は 11_linux-3.15.0 とする．スクリプトの記述例を以下に示す．

```
1 #!/bin/sh -e
2 echo "Adding my custom Linux to GRUB2"
3 cat << EOF
4 menuentry "My custom Linux" {
5 set root=(hd0,1)
6 linux /vmlinuz-3.15.0-linux ro root=/dev/sda2 quiet
7 initrd /initrd.img-3.15.0
8 }
9 EOF
```

スクリプトに記載された項目について以下に示す．

(a) menuentry < 表示名 >

< 表示名 > : カーネル選択画面に表示される名前

(b) set root=(< HDD 番号 >,< パーティション番号 >)

< HDD 番号 > : カーネルが保存されている HDD の番号

< パーティション番号 > : HDD の/boot が割り当てられたパーティション番号

(c) linux < カーネルイメージのファイル名 >

< カーネルイメージのファイル名 > : 起動するカーネルのカーネルイメージ

(d) ro < root デバイス >

< root デバイス >：起動時に読み込み専用でマウントするデバイス

(e) root=< ルートファイルシステム > < その他のブートオプション >

< ルートファイルシステム >：/root を割り当てたパーティション

< その他のブートオプション >：quiet はカーネルの起動時に出力するメッセージを省略可能

(f) initrd < 初期 RAM ディスク名 >

< 初期 RAM ディスク名 >：起動時にマウントする初期 RAM ディスク名

（B）実行権限の付与

/etc/grub.d で以下のコマンドを実行し，作成したスクリプトに実行権限を付与する．

```
$ sudo chmod +x 11_linux-3.15.0
```

（C）エントリ追加用のスクリプトの実行

以下のコマンドを実行し，作成したスクリプトを実行する．

```
$sudo update-grub
```

実行後，/boot/grub/grub.cfg にシステムコールを追加したカーネルのエントリが追加される．

(8) 再起動

以下のコマンドを実行し，計算機を再起動する．

```
$ sudo reboot
```

再起動後，GRUB2 のカーネル選択画面にエントリが追加されている．手順 (7) のスクリプトの例を用いた場合には，My custom Linux という名前のエントリが追加される．これを選択し，起動する．

6 テスト

カーネルのメッセージバッファに任意の文字列を出力するシステムコールが追加できているか否かテストする．このため，テストを行うプログラム test.c を作成する．

以下に test.c を記述する．

```
1 #include<unistd.h>
2
3 int main(){
4       char str[] = "Hello World";
5       syscall(317, str);
6       return 0;
7 }
```

このプログラムをコンパイルし実行する．カーネルのメッセージバッファに Hello World が表示されれば，システムコールが追加できている．以下のコマンドで，カーネルのメッセージバッファを確認する．

```
$ dmesg
```

以下のメッセージが表示されればシステムコールが正しく追加されている．

```
[ 3883.771978] Hello World
```

なお，左の数字は計算機の起動からの経過時間である．そのため，テストをする上では無視しても良い．

7 おわりに

本手順書では，カーネルのメッセージバッファに任意の文字列を出力するシステムコールを例にシステムコールの追加手順を示した．また，追加できたか否か確認するための手順を示した．

8 付録

付録にカーネルのメッセージバッファに任意の文字列を出力するシステムコールのソースコード例を添付する．

```
1 #include<linux/kernel.h>
2 #include<linux/syscalls.h>
3
4 asmlinkage void sys_mysyscall(char *str){
5         printk("%s\n", str);
6 }
```