システムコール実装の手順書

2018/4/10

藤原 裕貴

1 はじめに

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の実装手順を説明する．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．システムコールを実装するためのカーネルの取得から説明を行う．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 4 章 Linux カーネルの取得
第 5 章 システムコールの実装
第 6 章 テスト
第 7 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	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
```

エディタが起動し，"root ALL=(ALL) ALL"の直後に"fujiwara-yu ALL=(ALL) ALL" を追加する．なお，fujiwara-yu はユーザ名であり，自身のユーザ名に変更して使用する．

3.2 パッケージのインストール

以下に実装時に必要となるパッケージのインストール方法を git を例に示す．git はパッケージ名である．別のパッケージも同じ手順でインストールが可能である．

```
$ 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) libncurses5-dev
(4) bc
(5) make

4 Linux カーネルの取得

4.1 Linux のソースコードの取得

Linux のソースコードを Git から取得する．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．

```
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．なお，fujiwara-yu はユーザ名であり，自身のユーザ名に変更して使用すること．/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 以下にディレクトリが作成される．その後 cd コマンドにより，git ディレクトリに移動する．そのディレクトリで，git clone コマンドでリポジトリの複製を行うことで/home/fujiwara-yu/git 以下に linux-stable ディレクトリが作成される．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．ブランチとは開発の履歴を管理するための分岐である．/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 システムコール実装の手順

5.1 ソースコードの編集

システムコールを実装するために，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，追加した行の行頭に + を付け，削除した行の行頭には − を付けて表す．

(1) システムコール本体の実装
/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．以下に実装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】 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に対応した番号を定義する．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，この番号を引数することでシステムコールを呼び出すこ

とができる．また，システムコール番号は重複して定義することはできない．

```
  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
/home/fujiwara-yu/git/linux-stable/kernel/Makefile を以下のように編集し, 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 のコンパイルを追加する．

```
   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

以下の手順でカーネルの再構築を行う．/home/fujiwara-yu/git/linux-stable 以下で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
Linux をコンパイルする．以下のコマンドを実行する．

```
$ make bzImage -j8
```

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のことである．

(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 にコピーされる．

(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．以下のコマンドを実行する．

```
$ 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
```

これは，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次の手順で指定するため，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．初期 RAM ディスクとは，初期ルートファイルシステムのことである．初期 RAM ディスクは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．以下のコードを実行する．

```
$ 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．作成された初期 RAM ディスクの最後の 3.15.0 は与えた引数によって名前が決まる．

(7) ブートローダの設定

システムコールを実装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．/boot/grub/grub.cfg が設定ファイルである．本環境では設定ファイルを直接編集しない．使用されているブートローダは GRUB2 である．そのため，/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．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，既存のファイル名に倣い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する

(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実装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，計算機を再起動させる．

```
$ 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 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実装できているかテストする．このため，テストを行うプログラム test.c を /home/fujiwara-yu/git/linux-stable/ 以下に作成する．

以下に test.c を記述する．

```
1 #include<unistd.h>
2
3 #defile SYS_MYSYSCALL 317
4
5 int main(){
6       char str[] = "Hello World";
7       suscall(SYS_MYSYSCALL, str);
8       return 0;
9 }
```

このプログラムをコンパイルし実行する．実行後，以下のコマンドを実行することでカーネルのメッセージバッファを確認し，文字列が出力されているかを示す．

```
$ 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実装されている．

```
[ 3883.771978] Hello World
```

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実装手順を示した．また，実装の確認手順を示した．