資料タイトル

2018/4/10
吉田 修太郎

1 はじめに

本手順書では，Linux カーネルに対して新たにシステムコールを追加するための手順について述べる．本手順書において実装するのは，任意の文字列をカーネルバッファに書き込む機能を持つシステムコールである．以降では，実装環境，実装したシステムコールの概要，実装の手順，動作テストについて，順に章立ててそれぞれの詳細を述べる．

2 実装環境

本手順書における実装環境を下表に示す．

表 1　実装環境

OS	Debian 7.11
カーネル	Linux カーネル 3.15.0
CPU	Intel(R) Core(TM) i7-4770
メモリ	16GB

3 実装するシステムコールの概要

ここでは，本手順書において実装するシステムコールの概要について述べる．

形式　asmlinkage int sys_prt_to_rbuf(char *s)
引数　char *s: 出力する文字列のポインタ
戻り値　カーネルバッファに書き込んだ文字数
機能　引数として受け取った文字列をカーネルバッファに書き込む

4 実装の手順

4.1 概要

本章では，システムコール実装の手順について述べる．以降では，ソースコードの作成，プロトタイプ宣言，システムコール番号の定義，Makefile の編集およびカーネルの再構築についてそれぞれについ

て節を設けて詳細に述べる．

4.2 ソースコードの作成

ここでは，実装するシステムコールのソースコードの作成について述べる．具体的な記述例として，本手順書 3 章において示したシステムコールの，ソースコードを以下に示す．なお，このソースコードは C 言語で記述されている．

```
#include <linux/kernel.h>
#include <linux/syscalls.h>

asmlinkage int sys_prt_to_rbuf(char *s){
  int ret;
  ret = printk(KERN_INFO "%s\n",s);
  printk("%d charactor(s) outputed\n",ret);
  return ret;
}
```

4.3 プロトタイプ宣言

ここでは，プロトタイプ宣言の手順について述べる．システムコール関数のプロトタイプ宣言が，まとめて書かれているヘッダファイルを探し，編集する．本手順書では，以下のファイルを編集する．

- /home/git/linux-stable/include/linux/syscalls.h

本手順書では，このヘッダファイルの末尾に以下の行を追加する．

- /home/git/asmlinkage int prt_to_rbuf(char *s);

4.4 システムコール番号の定義

ここでは，システムコール番号を定義する手順について述べる．システムコール関数と，システムコール番号との対応づけが書かれているファイルを探し，編集する．本手順書では，以下のファイルを編集する．

- /home/git/linux-stable/arch/x86/syscalls/syscall_64.tbl

このファイルの内容の一部を抜粋し，以下に示す．

```
#
# 64-bit system call numbers and entry vectors
```

```
#
# The format is:
# <number> <abi> <name> <entry point>
#
# The abi is "common", "64" or "x32" for this file.
#
0 common read sys_read
1 common write sys_write
2 common open sys_open
3 common close sys_close
               ~ (中略) ~
314 common sched_setattr sys_sched_setattr
315 common sched_getattr sys_sched_getattr
316 common renameat2 sys_renameat2

#
# x32-specific system call numbers start at 512 to avoid cache impact
# for native 64-bit operation.
#
512 x32 rt_sigaction compat_sys_rt_sigaction
513 x32 rt_sigreturn stub_x32_rt_sigreturn
514 x32 ioctl compat_sys_ioctl
               ~ (中略) ~
540 x32 process_vm_writev compat_sys_process_vm_writev
541 x32 setsockopt compat_sys_setsockopt
542 x32 getsockopt compat_sys_getsockopt
(EOF)
```

上記のファイル内容の先頭付近に，このファイルのフォーマットは< number >< abi >< name >< entry point >であるという旨が記述されている．それぞれの要素についての簡単な説明を以下に示す．

number　システムコール番号
abi　Applicatioin Binary Interface
name　関数名
entry point　関数が

このフォーマットに従い，実装したいシステムコールをこのファイルに追記する．ただし，システムコール番号は，システムコール呼出しの際に関数の特定に使用されるため，他の関数と重複してはなら

ない．本資料では，以下のように設定する．

number　317
abi　common
name　sys_prt_to_rbuf
entry point　sys_prt_to_rbuf

上記の場合におけるファイルへの記入例 (変更した部位とその前後数行のみ抜粋) を以下に示す．先頭に + を置いて示した行が，追加された行である．

```
 314 common sched_setattr sys_sched_setattr
 315 common sched_getattr sys_sched_getattr
 316 common renameat2 sys_renameat2
+317     commoc  sys_prt_to_rbuf         sys_prt_to_rbuf

#
# x32-specific system call numbers start at 512 to avoid cache impact
# for native 64-bit operation.
#
```

4.5　Makefile 編集

ここでは，Makefile の編集について述べる．今回編集する Makefile を以下に示す．

- /home/git/linux-stable/kernel/Makefile

make コマンドは，このファイルの内容に基づいて実行されるため，今回追加したシステムコール関数をコンパイルするためには，ここにその処理を追記する必要がある．具体的には，Makefile の先頭付近に記述されている，各システムコール関数のオブジェクトファイルが代入される obj-y という変数に対して，新たに作成したシステムコールのオブジェクトファイルも代入されるように追記する．本手順書における，この Makefile の編集内容 (編集した部分のみ抜粋) を以下に示す．なお，ここでは削除した行の先頭に-を，追加した行の先頭に＋を挿入している．

```
 #
 # Makefile for the linux kernel
 #

obj-y     = fork.o exec_domain.o panic.o \
            cpu.o exit.o itimer.o time.o softirq.o resource.o \
            sysctl.o sysctl_binary.o capability.o ptrace.o timer.o user.o \
```

```
              signal.o sys.o kmod.o workqueue.o pid.o task_work.o \
              extable.o params.o posix-timers.o \
              kthread.o sys_ni.o posix-cpu-timers.o \
              hrtimer.o nsproxy.o \
              notifier.o ksysfs.o cred.o reboot.o \
-             async.o range.o groups.o smpboot.o
+             async.o range.o groups.o smpboot.o   prt_to_rbuf.o
```

4.6 カーネルの再構築

4.6.1 .config ファイルの作成

カーネルを再構築するにあたり，はじめに，.config ファイルを作成する．これは，カーネルの設定ファイルである．

4.6.2 カーネルのコンパイル

4.6.3 カーネルのインストール

4.6.4 カーネルモジュールのコンパイル

4.6.5 カーネルモジュールのインストール

5 動作テスト

5.1 概要

本章では，実装したシステムコールの動作テストについて述べる．以降では，動作テスト用プログラムの準備と，動作テストの手順について，順に節を設け述べる．

5.2 動作テスト用プログラムの準備

prt_to_rbuf の動作テストに用いるプログラムを作成する．本手順書における動作テスト用プログラムを以下に示す．

```
#include<stdio.h>
#include<unistd.h>

int main(){
  char buf[128];
  long sys_num = 317;
  int ret;
  scanf("%[^\r\n]",buf);
```

```
  ret = syscall(sys_num,buf);
  printf("ret:%d\n",ret);
  return 0;
}
```

5.3 動作テストの手順

動作テストの手順を以下に示す.

(1) 動作テスト用プログラムの実行
(2) dmesg コマンドの実行

6 おわりに

本資料では