Linuxカーネルに対するシステムコール追加のための手順書

2018/4/13

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．本手順書で追加するシステムコールは，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．なお，本手順書はコンソールの基本操作と `Git` の基礎知識を習得している者を対象としている．以降では，追加するシステムコールの実装環境，追加するシステムコールの概要，システムコールの追加手順，システムコールの動作テストについて，順に章立てて詳細を述べる．

2 追加するシステムコールの実装環境

本手順書において追加するシステムコールの実装環境を表 1 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 システムコールの追加手順

3.1 概要

本章では，新たにシステムコールを追加する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，環境設定，Linux カーネルの入手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．また，本章において何らかのファイルの内容を編集する際には，追加した行の先頭に + を，削除した行の先頭に-を記して示す．

3.2 ソースコードの作成

ここでは，追加するシステムコールのソースコードを作成する．本手順書で追加するシステムコールの概要は以下のとおりである．

追加するシステムコールの概要

形式　asmlinkage int sys_prt_to_rbuf(char *s)

引数　char *s: 任意の文字列を指すポインタ

戻り値　カーネルのメッセージバッファに書き込んだ文字数

機能　任意の文字列をカーネルのメッセージバッファに書き込む

3.3 関数のプロトタイプ宣言

本節では，関数のプロトタイプ宣言を行う．本手順書では，以下のファイルを編集し，プロトタイプ宣言を行う．

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

本手順書では，上記のファイルを以下で示すように編集する．

```
 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 int prt\_to\_rbuf(char *s);
```

編集により，prt_to_rbuf のプロトタイプ宣言がなされる．

3.4 システムコール番号の定義

本節では，システムコール番号を定義する．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
```

上記のファイルを以下に示すように編集する．

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

編集により，317 番が sys_prt_to_rbuf のシステムコール番号として定義された．

3.5 Makefile の編集

ここでは，Makefile を編集する．今回編集する Makefile を以下に示す．

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

本手順書では，上記ファイル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
+14              async.o range.o groups.o smpboot.o prt_to_rbuf.o
```

編集により，prt_to_rbuf.c が，カーネルのコンパイルに含まれる．

3.6 カーネルの再構築

3.6.1 概要

本節では，カーネル再構築の手順について述べる．なお，以降で現れる username は，操作を行う計算機にログインしているユーザ名である．

3.6.2 環境設定

(1) sudo 権限の付与

以降の作業を行うユーザに sudo 権限を付与する．以下のコマンドを実行する．

```
$ su
# visudo
```

ここでエディタが起動し，sudoers.tmp が編集できるようになる．ここで，sudoers.tmp を以下のように編集する．

```
 19 # User privilege specification
 20 root    ALL=(ALL:ALL) ALL
+21 username ALL=(ALL) ALL
```

編集により，username に sudo 権限が付与される．

(2) git と gcc のインストール

git と gcc をインストールする．以下のコマンドを実行する．

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

実行により，git と gcc がインストールされる．

(3) ライブラリとコマンドのインストール

カーネル再構築で使用するライブラリとコマンドをインストールする．以下のコマンドを実

行する．

```
$ sudo apt-get install libncurses5-dev
$ sudo apt-get install bc
```

実行により，必要なライブラリとコマンドがインストールされる．

3.6.3 Linux カーネルの入手

ソースコードの入手

Linux のソースコードを入手する．Linux のソースコードが管理されている，Git のリポジトリをクローンする．本手順書では，/home/username 以下に git ディレクトリを設け，/home/username/git 以下でクローンを行う．以下のコマンドを実行する．

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

実行により，/home/username/git 以下に，Linux のソースコードが格納された linux-stable ディレクトリが作成される．

ブランチの作成と切替え

ブランチの作成と切替えを行う．本手順書で使用する Linux のバージョンは 3.15 なので，そのバージョンのコミットへ移動し，新規のブランチを作成，移動する．以降で示される操作は，すべて/home/username/git/linux-stable 以下で行う．以下のコマンドを実行する．

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

実行により，v3.15 というタグが示すコミットから新規のブランチ 3.15 が作成され，ブランチ 3.15 に切り替わる．

3.6.4 .config ファイルの作成

.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．以下のコマンドを実行する．

```
$ make defconfig
```

実行により，x86_64_defconfign ファイルに基づき，.config ファイルが/home/username/git/linux-stable 以下に生成される．

3.6.5 カーネルのコンパイル

システムコールを追加するカーネルをコンパイルする．以下のコマンドを実行する．

```
$ make bzImage -j8
```

実行により，/home/username/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，/home/username/git/linux-stable 以下に System.map というファイルが生成される．上記コマンドに設定されている-j オプションでは，同時に実行できるジョブ数の上限を指定している．これは，同時に実行するジョブ数が増え過ぎると，実行速度が低下する恐れがあるためである．また，このときのジョブ数は，CPU のコア数*2 程度が効果的であるとされる [1].

3.6.6 カーネルのインストール

コンパイルしたカーネルのインストールを行う．以下のコマンドを実行する．

```
$ sudo cp /home/username/git/linux-stable/arch/x86/boot/bzImage /boot/vmlinuz-3.15.
$ sudo cp /home/username/git/linux-stable/System.map /boot/System.map-3.15.0-linux
```

実行により，カーネルをコンパイルした際に生成された，bzImage と System.map が，vmlinuz-3.15.0-linux と System.map-3.15.0-linux として/boot 以下にコピーされ，カーネルのインストールが完了する．

3.6.7 カーネルモジュールのコンパイル

カーネルモジュールをコンパイルする．以下のコマンドを実行する．

```
$ make modules
```

上記コマンドの実行により，カーネルモジュールがコンパイルされる．

3.6.8 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のインストールする．以下のコマンドを実行する．

```
$ make modules_install
```

上記コマンドの実行により，コンパイルしたカーネルモジュールがインストールされる．このとき，端末に実行結果が出力されるが，その最下行には以下のような表記がある．

DEPMOD 3.15.0+

この，DEPMOD 以後に続く 3.15.0 ＋という部分は，カーネルモジュールがインストールされたディレクトリ名であり，環境によって異なる．このデレクトリ名は次の操作で用いるため，控えておく．

3.6.9 初期 RAM ディスクの作成

カーネルの再構築において，本項以降の操作は，対象となるカーネルの初回の構築時のみ必要な手順である．初期 RAM ディスクを作成する．本指示書では，以下のコマンドを実行するが，このとき，3.6.8 で控えておいたディレクトリ名を引数として与える．

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

実行により，初期 RAM ディスクが作成される．

3.6.10 ブートローダの設定と再起動

ブートローダの設定と再起動を行う．本手順書における環境では，ブートローダは GRUB2 が用いられている．GRUB2 の設定ファイルは/boot/grub/grub.cfg であるが，このファイルは gurb によって生成され，直接編集してはいけない．/etc/grub.d 以下にあるファイルを基に，この設定ファイルが生成されるので，このディレクトリにスクリプトを追加した後，コマンドを用いて設定ファイルを書き換える．その後，計算機をシステムコールを追加したカーネルで再起動する．手順を以下に述べる．

(1) 追加するスクリプトの作成
追加するスクリプトを作成する．スクリプトのファイル名であるが，本手順書では/etc/grub.d にある README ファイルの説明に沿うように 11_linux-3.15.0 とする．このスクリプトの内容を以下に示す．

```
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
```

4 行目 mementry ”My custom Linux”
ブートローダのカーネル選択画面に表示される名前を設定している．ここでは ”My custom Linux ” としている．

5 行目 set root=(hd0,1)
root の値を設定している．ここで設定した場所に/boot が割り当てられる．hd0 は HDD の番号，その右の 1 はパーティション番号である．

6 行目 linux /vmlinuz-3.15.0-linux ro root=/dev/sda2 quiet

linux /vmlinuz-3.15.0-linux
指定したファイルから，Linux カーネルイメージを読み込んでいる．ここでは 3.6.6 でインストールしたファイルを指定している．
ro root=/dev/sda2

ルートファイルシステムと，そのマウント方法を指定している．

quiet

カーネル起動時におけるメッセージ出力の省略を指定している．

7 行目 initrd /initrd.img-3.15.0

読み込む初期 RAM ディスクを指定している．

(2) 実行権限の付与

作成したスクリプトに実行権限を付与する．以下のコマンドを実行する．

```
$ sudo chmod +x /etc/grub.d/11_linux-3.15.0
```

実行により，11_linux-3.15.0 に実行権限が付与される．

(3) ブートローダ設定ファイルの再生成

ブートローダの設定ファイルを再生成する．以下のコマンドを実行する．

```
$ sudo update-grub
```

実行により，ブートローダの設定ファイルが再生成される．そして，システムコールを追加したカーネルで起動できるようになる．

(4) 再起動

再起動を行う．以下のコマンドを実行する．

```
$ sudo reboot
```

実行により，計算機が再起動する．このとき，カーネル選択画面で My custom Linux がエントリに追加されているので，それを選択し，起動する．

4 動作テスト

4.1 概要

本章では，追加したシステムコールの動作テストについて述べる．プログラムを用いて，追加したシステムコールを呼び出し，その振る舞いが意図したものであるか確認する．以降では，動作テスト用プログラムの作成と，動作テストの手順について，順に節を設け述べる．

4.2 動作テスト用プログラムの作成

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．

```
1 #include<stdio.h>
2 #include<unistd.h>
3
```

```
 4 int main(){
 5  char buf[128];
 6  long sys_num = 317;
 7  int ret;
 8  scanf("%[^\r\n]",buf);
 9  ret = syscall(sys_num,buf);
10  printf("ret:%d\n",ret);
11  return 0;
12}
```

上記の動作テスト用プログラムは, 端末から任意の文字列の入力を受けつけ, syscall に第 2 引数として渡す．同時に, syscall には 317 という値が第 1 引数として渡されているが, これは sys_prt_to_rbuf のシステムコール番号である．syscall によって sys_prt_to_rbuf を呼び出し，戻り値を端末に出力して終了する.

4.3 動作テストの手順

追加したシステムコールの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

まず動作テストプログラムを実行する．本手順書における動作テスト用プログラムは，実行時に任意の文字列の入力を要求する．今回は，”under the water”と入力する．このとき，追加したシステムコールが適切に動作していれば，カーネルのメッセージバッファに先ほどのの文字列”under the water”が書き込まれているはずである．そこで，dmesg コマンドを用いる．dmesg コマンドは，カーネルのメッセージバッファの内容を端末に出力する機能を持つ．このときの出力結果を以下に示す.

```
[12300.934433] under the water
[12300.934434] 18 charactor(s) outputed
```

上記の結果により，指定した文字列”under the water”がカーネルバッファに書き込まれていることがわかる．なお，各ログの左に表示されている数字は，計算機が起動してから各ログがメッセージバッファに書き込まれた時点までに経過した秒数である．追加したシステムコールは，意図した動作をしている.

5 おわりに

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手順について述べた．さらに，追加したシステムコールが，意図した動作をしているかどうかを調べるテストの手順について述

べた.

参考文献

[1]