システムコール追加の手順書

2020/5/27

松田 陸斗

1　はじめに

本手順書では，Linux カーネルへシステムコールを追加する手順を記す．カーネルへシステムコールを追加するには，Linux のソースコードを取得してシステムコールのソースコードを追加した後，カーネルの再構築を行う．また，本手順書で想定する読者はコンソールの基本的な操作を習得しているものとする．以下に本手順書の章立てを示す．

1 章　はじめに
2 章　追加環境
3 章　追加したシステムコール
4 章　システムコールの追加の手順
5 章　テスト

2　追加環境

システムコールを追加する環境を表 1 に示す．

表 1　実行環境

項目名	環境
OS	
カーネル	
CPU	
メモリ	

3　追加したシステムコール

本実験では，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追加した．

形式　`asmlinkage void sys_my_syscall(char *msg)`
引数　`char *msg`：任意の文字列
戻り値　なし
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

4　システムコールの追加の手順

Linux カーネルの再構築の手順を以下に示す．

4.1　環境設定

4.1.1　APT のリポジトリの設定

パッケージのダウンロード元設定ファイルである sources.list 編集し，ダウンロード元として cdrom を削除する．

```
$ su -
# vi /etc/apt/sources.list
```

4.1.2　sudo 権限の付与

sudo をインストールし，ユーザに sudo 権限を与える．以下のコマンドを実行する．

```
# apt update
# apt install sudo
# visudo
```

4.1.3　git と gcc のインストール

git，gcc，および make をインストールする．以下のコマンドを実行する．

```
$ sudo apt update
$ sudo apt install git gcc make
```

4.2　Linux カーネルの取得

4.2.1　Linux のソースコードの取得

Linux のソースコードを取得する．Linux のソースコードは Git で管理されている．Git とは，オープンソースの分散型バージョン管理システムである．リポジトリとは，ディレクトリを保存する場所のことであり，クローンとは，リポジトリの内容を任意のディレクトリに複製することである．本手順書では，/home/matsuda/git 以下でソースコードを管理する．/home/matsuda 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
```

実行後，mkdir コマンドにより/home/matsuda 以下に git ディレクトリが作成される．そして，cd コマンドにより，git ディレクトリに移動する．git clone コマンドにより，/home/matsuda/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．

4.2.2 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/matsuda/git/linux-stable で以下のコマンドを実行する．

```
$ git checkout -b 4.19 v4.19
```

実行後，v4.19 というタグが示すコミットからブランチ 4.19 が作成され，ブランチ 4,19 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．

4.3 カーネルの再構築

4.3.1 .config ファイルの生成

.config ファイルを以下のコマンドで生成する．

```
make defconfig
```

.config ファイルとは，カーネルの設定を記述したコンフィギュレーションファイルである．実行後，/home/user/git/linux-stable 以下に.config ファイルが生成される．

4.3.2 カーネルのコンパイル

Linux カーネルを以下のコマンドでコンパイルする．

```
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．ジョブ数は CPU のコア数*2 が効果的である．コマンド実行後，/home/matsuda-ri/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/matsuda-ri/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．

4.3.3 カーネルのインストール

コンパイルしたカーネルを以下のコマンドでインストールする．

```
$ sudo cp arch/x86/boot/bzImage /boot/vmlinuz-4.19.0-linux
$ sudo cp System.map /boot/System.map-4.19.0-linux
```

実行後，bzImage と System.map が，/boot 以下にそれぞれ vmlinuz-3.16.0-linux と System.map-3.16.0-linux という名前でコピーされる．

4.3.4 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以下のコマンドでコンパイルする．

```
$ make modules
```

カーネルモジュールとは，機能を拡張するためのバイナリファイルである．

4.3.5 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以下のコマンドでインストールする．

```
$ sudo make modules_install
```

実行結果の最後の行は以下のように表示される．

```
DEPMOD 4.19.0
```

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を表示している．

4.3.6 初期 RAM ディスクの作成

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．これは，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．以下のコマンドを実行する．

```
$ sudo update-initramfs -c -k 4.19.0
```

4.3.7 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に，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．エントリを追加するためには，このファイルを編集する必要がある．Debian10 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．ブートローダを設定する手順を以下に示す．

(1) 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-4.19.0 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-4.19.0-linux ro root=/dev/sda3 quiet
7 initrd /initrd.img-4.19.0
8 }
9 EOF
```

(2) 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-4.19.0
```

(3) 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．

4.3.8 再起動

任意のディレクトリで以下のコマンドを実行し，計算機を再起動させる．

```
$ sudo reboot
```

GRUB2 のカーネル選択画面にエントリが追加されている．カーネル選択画面で My custom Linux を選択し，起動する．

5 テスト

5.1 概要

本手順書で追加したシステムコールが正常に動作しているかを確認するためにテストを行う．

5.2 テストプログラムの作成

追加したシステムコールを呼び出すテストプログラムを作成する．テストプログラムの処理の流れは以下の通りである．

(1) 標準入力から文字列を受け取る
(2) 受け取った文字列を引数として，追加したシステムコールを呼び出す

5.3 テストプログラムの実行

テストプログラムをコンパイルし実行する．

5.4 カーネルバッファの内容を確認

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
```

実行結果として以下の結果が得られる．

研究室で確認

この結果より，カーネルバッファに任意の文字列を出力するシステムコールが作成できたことがわかる．