2-4 ハードウェア・アクセスや割り込み処理に関わる障害の解析

宗像 尚郎

⑲ ハードウェア・アクセスに起因する障害の解析

● ハードウェアの制御はカーネルのデバイス・ドライバに「依頼」する

第2章 2-3で，プロセス内で動作するプログラムは，全てのハードウェア資源を占有していると思い込んでいると説明しました．**図1**のようにプロセスは仮想メモリ空間に配置されているので，物理アドレス上に配置されたハードウェアには直接アクセスできません．ハードウェアを利用するときは，カーネルにアクセス代行を依頼する必要があります．

マイコンでは，ポート出力をソフトウェア的に変化させてLEDをON/OFFさせるLチカ・プログラムが入門の定番として有名ですが，Linux環境でLチカを実現するのは相当大変なことなのです．物理ハードウェアの制御は，カーネル内のデバイス・ドライバの専権事項です．ユーザ空間のプログラムからデバイスを利用するときには，プログラムの実行をいったんカーネル空間に切り替えて，ドライバ経由でハードウェア・アクセスを行います．第2章 2-3でハード・ディスクへアクセスしたときにsys時間をたくさん消費したのは，ハード・ディスクへのアクセスをカーネルのデバイス・ドライバに依頼していたからです．

● 依頼時に使う低水準関数「システム・コール」

C言語プログラムでハード・ディスクからデータを読むときは，`fopen()`でファイル・ディスクリプタを生成してから，`fgetc()`/`fgets()`でデータをリードします．ここで使った`fopen()`などは，glibcに組み込まれたライブラリ関数です．このプログラムをコンパイルしてLinux上で実行すると，カーネルにデバイス・アクセスの代行依頼が発行されています．そのときに使われるのが，POSIX（Portable Operating System Interface）によって規定されたシステム・コールの`open()`や`read()`などの低水準関数です．

Linuxでは，カーネルのバージョンによって利用できるシステム・コールの種類に違いがあるので，利用可能なコンパイラやglibcのバージョンがカーネルのバージョンごとに指定されています．`man syscalls`と入力すると，現在実行しているLinux環境で利用可能なシステム・コールを確認できます．プログラムの中で低水準関数を使えば，明示的にシステム・コールの発行をコントロールできますが，ライブラリ関数を利用するのが一般的でしょう．

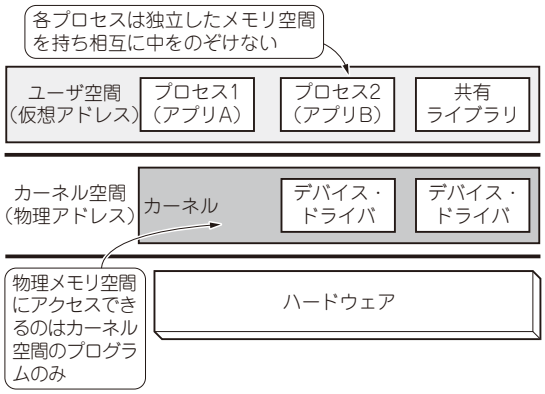

図1　プロセスとカーネルの関係
プロセス（アプリケーション）は仮想メモリ空間（ユーザ空間）で動くので，物理アドレス上に配置されたハードウェアに直接アクセスできない．ハードウェアを制御するときは，カーネル内のデバイス・ドライバに代行を依頼する

● システム・コールの発行状況を表示する「`strace`コマンド」

ここでは，主に次のような内容を表示できる`strace`コマンドを紹介します．

- プログラムが発行したシステム・コール
- 呼び出されたカーネル機能の実行にかかった時間
- 多く利用されたシステム・コールは何か

▶使い方

基本的な使い方は簡単です．次のように実行すると，コマンドが呼び出したシステム・コールを呼び出し順に表示します．

```
$ strace（実行コマンド）↵
```

`strace`を実行すると，かなりたくさんの情報が表示されるので，`-o`オプションを使って結果をファイルに書き出すと便利かもしれません．