第1章 C言語による疑似OSで体験

OSカーネルのマルチタスク制御の仕組み

坂井 弘亮

OSと通常のプログラムの違い

● 割り込みがある前提で動作する

OSカーネルの基本機能として，タスク管理があります．特にマルチタスクによるCPU資源の割り当てはOSカーネルの機能の中核とも呼べる重要なものですが，理解が難しい部分もあるように思います．

その大きな理由は，プログラム自身がプログラムを管理することにより，通常のアプリケーション・プログラムとは異なる考え方が必要なためと思います．

異なる考え方とは，割り込みをベースとした動作です．プログラム自身がプログラムを管理するということは，あるプログラムの実行中に，別のプログラムが割り込む必要があるということです．このためマルチタスクの実現には割り込み処理が重要なものとなってきます．

● 割り込み発生時の動作

▶通常のプログラムの場合

割り込みをベースとした動作とは，どのようなものでしょうか．通常のプログラムは，サブルーチン（C言語ならば，関数）を呼び出すと，呼び出し元に戻ってきます．**図1**はそのような関数呼び出しの例です．`func1()`から`func2()`が呼び出され，さらにそこから`func3()`と`func4()`が呼び出されています．このように関数呼び出しが多重に行われても，必ず呼び出し元である`func1()`に戻ってくることになります．

しかしOSカーネルのような割り込みベースのプログラムは違います．

▶OSカーネルの場合…割り込みを合図にタスクを切り替える

図2は典型的なOSカーネルの動作の例です．まず割り込みが入り，割り込みに応じた処理をします．割り込み処理を行う箇所は一般にISR（Interrupt Service Routine）と呼ばれます．

次に動作すべきタスクをスケジューリングし，そのようなタスクがあるならばそのタスクをディスパッチすることで，タスクの動作に移ります．動作できるタスクがないならば，CPUをスリープ状態にして割り込み待ちに入ります．

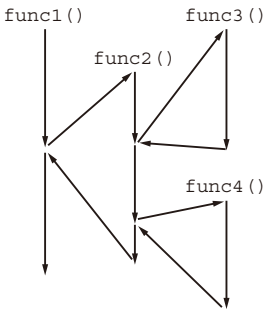

図1 サブルーチン・コールの例

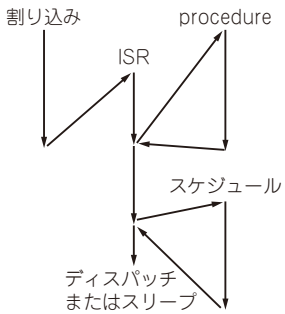

図2 OSカーネルの動作

いずれにしても，ISRから戻ってくることはありません．このためOSカーネルのプログラムは，「戻って来ない関数」というものが存在します．割り込みハンドラの延長で何らかの関数を呼び出しても，その先でタスク・ディスパッチが行われるため，戻って来ない，という関数がある訳です．OSカーネルのプログラムを読む際には，そうした「戻って来ない関数がある」という認識が必要です．

完全C言語の疑似OSでカーネルのタスク制御を見える化

● 割り込みライクなプロセス間通信「シグナル」を割り込みに見立てる

UNIXや他システム上で動作するアプリケーションで，割り込みに似せたものとして「シグナル」があります．シグナルはアプリケーションの実行とは非同期に発生し，シグナル・ハンドラが呼ばれる，というものです．この中で行う処理には排他制御やリエントラント（再入可能）性の配慮などが必要となるため，シグナル・ハンドラではフラグを立てるだけなどの簡単な処理にとどめ，処理の本体はアプリケーション側でフラグを見て行う，というような設計にするのが普通です．

しかし，シグナルを利用すれば実行中のプログラムに対しての割り込みができる訳ですから，原理的には