■リエントラント性とスレッドセーフ

●Linuxライターの作業手順とリエントラント性

突然ですが、「組込みプレス」と「Software Design」という２種類の
雑誌があります。
通常はこれら各々の読み手となるターゲットは異なるのですが、
たまたまどちらの雑誌もLinuxの特集を行うことになったとします。
さらに、両方の編集部から一人のLinuxを得意とするライターに執筆の
依頼が届いたとします。
大雑把にはLinuxというカテゴリの記事ですが、それぞれ詳細な
記事の目的は異なります。
締め切り日も近く、どうしても並行して執筆作業をする必要がありましたが、
どちらの仕事もそのライターは引き受けてしまいました。

そのLinuxライターは一人、執筆に使う道具も同じです。
ですが、それぞれの記事はそれぞれの依頼元の意向に沿った形で執筆し、
また、提出する原稿の内容はそれぞれ独立している必要があります。

もし同じ原稿用紙に１行ごとにそれぞれの雑誌の記事を書き込んでいく
などということをしたらどうなるでしょうか。
記事の内容としてはとても脈絡のないものになってしまい、依頼元の
要求どおりの出力は得られないことでしょう。

一人のライターがそれぞれの雑誌の記事をそれぞれの意向に沿った形で
提出できる状態がリエントラント性を持った処理であるということが
言えます。

●複数のアプリケーションから同じ処理を呼び出す

ソフトウェアの開発に話を戻します。
ここで、ライターを共通のプログラム、依頼元の意向をプログラムに
対しての引数、独立した出力結果を得る為に内部で使用するメモ用紙を
独立したリソース、出力結果もまた独立したリソースと読み替えて見ます。

つまり、プログラム実行コードの実体は一つしかなくとも、
それぞれの引数を独立して解釈でき、それに応じたリソースを
使用して、複数の独立した結果を出せることが「リエントラント性がある」
ということが出来ます。

それぞれの処理の手続きの流れを以下に示します。

＜組込みプレス出版処理＞
①プログラム(kpress.exe)開始
②処理Aを行う
③サブルーチンLinuxWriter(組込みプレス用執筆依頼)を呼び出す
④出力結果を貰う
⑤処理Bを行う
⑥プログラム(kpress.exe)終了

＜Software Design出版処理＞
①プログラム（sdesign.exe）開始
②処理αを行う
③サブルーチンLinuxWriter(Software Design用執筆依頼)を呼び出す
④出力結果を貰う
⑤プログラム(sdesign.exe)終了

最初のプログラムkpress.exeと次のプログラムsdesign.exeは共に
同じ手続きを処理するサブルーチンLinuxWriter()を呼び出しています。

●共通のプログラムはどこに実体があるか

ここで、２つのプログラムを「プロセス」と表現しなおしましょう。
プロセスの場合、kpress.exeとsdesign.exeの両方のプロセスの中に
LinuxWriter()がクローンとしてそれぞれに実体が存在し、静的な
メモリはもちろんのこと、スタック上の変数などは独立した
仮想メモリ空間に存在できます。
この時、LinuxWriter()はスレッドセーフである必要はありません。

ところが「技術評論社プロセス」という一つのプロセスが
「組込みプレス出版スレッド」と「SoftwareDesign出版スレッド」を
生成、実行してLinuxWriter()を使用する場合はLinuxWriter()は
スレッドセーフである必要があります。

では、この２つのプログラムがスレッドや仮想記憶をサポートしない
RTOS上で動作するタスクだったとします。
同様に呼び出されるLinuxWriter()はスレッドセーフである必要が
あります。
更に、LinuxWriter()も独立したタスクとして扱うことにすると、
「スレッドセーフ」という表現は使わず、「リエントラント性がある」
という言葉を使います。

●まとめ
最後にリエントラント性とスレッドセーフについて整理します。

・プロセス単位で使う共通処理は必ずしもスレッドセーフである必要はない。
・同じメモリ空間を共有しているマルチタスク環境や同じプロセスから生成される
  複数のスレッドから呼び出させる共通処理はスレッドセーフである必要がある。
・スレッドセーフという言葉はプログラムの一部に存在する共通処理部分に対して
  使われる。
・リエントラント性という言葉は抽象的な話の中やタスクやスレッド自体が
  複数の処理依頼を同時に受け付けることが出来るような場合に使用する。
  ※最近はこのような場合でもスレッドセーフを使う事も多い。