日曜研究室

技術的な観点から日常を綴ります

[xv6 #38] Chapter 3 – Locking – Lock ordering

テキストの47ページ

本文

カーネルを通るコードパスが、いくつかのロックを取らなければならない場合、全てのコードパスが同じ順番でロックを獲得することが重要である。
そうしなければ、デッドロックのリスクがある。
xv6の2つのコードパスが、ロックAとロックBを必要とするとしよう。
しかしコードパス1は、A→Bの順番でロックを獲得し、コードパス2はB→Aの順番でロックを獲得するとする。
この場合、コードパス1はロックBを獲得する前にロックAを獲得し、コードパス2はロックBを獲得するかもしれないので、デッドロックに至る可能性がある。
コードパス1は、コードパス2が保持しているロックBを必要とし、コードパス2は、コードパス1が保持しているロックAを必要とするので、そうなったらどちらのコードパスも続行することが出来なくなる。
そのようなデッドロックを避けるため、全てのコードパスは、同じ順番でロックを獲得すべきである。
デッドロックを回避することは、ロックをなぜ関数の特性の一部と見なさなければならないかを、説明しているもう一つの例である。
呼び出し側は、その関数が同じ順番でロックを獲得するために、一貫した順番で関数を呼び出さなければならない。

xv6は粗い粒度のロックを使い、またxv6はシンプルではあるが、xv6はいくつかのロック順の連鎖を持つ。
しかし一番長くても、2連鎖である。
例えば、ideintr関数は、ptable.lockを必要とするwakeup関数を呼んでいる間、idelockを保持する。
sleep関数やwakeup関数を使っている例は、他にも若干ある。
sleep関数やwakeup関数は（第4章で説明するような）複雑なインバリアントを持っているので、それらの順序付けが生じる。
ファイルシステムでも、2連鎖の例がいくつかある。

例えば、あるファイルをその親ディレクトリから正しくアンリンクする場合、そのファイルのロックとディレクトリのロックを獲得しなければならないからである。
xv6は常に、最初に親ディレクトリ、そして次にファイル、という順番でロックを獲得する。

感想

デッドロックしないようにするために、ロックを獲得する順番が大事という話です。
ファイルシステムの話も少しだけ出てきましたが特に難しくはないですね。

カテゴリー: 技術 l タグ: xv6 l 投稿日: 2012/3/13 火曜日 [http://peta.okechan.net/blog/archives/1436] l