日曜研究室

技術的な観点から日常を綴ります

[xv6 #44] Chapter 4 – Scheduling – Multiplexing

テキストの51ページ

本文

xv6は、次の多重化方法を採用する。
あるプロセスがディスクへのリクエストを待っているとき、xv6はそのプロセスをスリープさせ、他のプロセスの実行をスケジュールする。
さらに、xv6はタイマ割り込みを使って、あるプロセッサ上で実行中のプロセスを一定時間（100msec）で強制的に停止させるので、そのプロセッサで他のプロセスの実行をスケジュールすることができる。
この多重化方法は、それぞれのプロセスが自分専用のCPUを持ってるように見せかける効果を生み出す。
これはちょうど、xv6がメモリアロケータとハードウェアページテーブルを使って、それぞれのプロセスが自分専用のメモリを持ってるように見せかける効果を生み出す事に似ている。

多重化の実装は、いくつかの難問を含んでいる。
ひとつめ、あるプロセスから他のプロセスにどうやって切り替えるか？
xv6は、コンテキストスイッチの標準的な仕組みを利用する。
そのアイデアはシンプルである、とはいえ、実装しなければならないコードは、OSを構成するコードの一番ややこしい部分の典型例である。
ふたつめ、どうやって透過的にコンテキストを切り替えるか？
xv6は、タイマ割り込みハンドラを利用してコンテキストスイッチを実行するという、標準的なテクニックを使う。
みっつめ、多くのCPUは多くのプロセス間で同時に切り替えを行うので、競合を避けるためにうまくロックを使うことが必要となる。
よっつめ、プロセスが終了したとき、そのメモリとその他のリソースは解放されなければならないが、（例えば）プロセスは自分自身のカーネルスタックをそれが使われている間は開放できないので、解放処理の全てをそのプロセス自身で行う事は出来ない。
xv6は以上の問題をできるだけ簡単に解決しようとしたが、それでもやはり結果的にはコードはトリッキーとなっている。

xv6は、プロセスにそれら同士で協調するための方法を提供しなければならない。
例えば、親プロセスがある子プロセスの終了を待つような場合や、あるプロセスがパイプを読み込むために他のプロセスがそのパイプに書き込むのを待つような場合に。
待っているイベントが起きたかどうか繰り返し調べることによって、待ってる側のプロセスがCPUを浪費するような方法ではなく、イベントを待つためにそのプロセスをスリープさせCPUを手放させ、他のプロセスがそのプロセスを再開する事が、xv6では可能となっている。
イベント通知のロスによって発生する競合を避けるために注意が必要である。
以上の問題と解決方法の例として、この章ではパイプの実装について説明する。

感想

ということみたいです。

カテゴリー: 技術 | タグ: xv6 | 投稿日: 2012/3/18 日曜日 [http://peta.okechan.net/blog/archives/1541] |