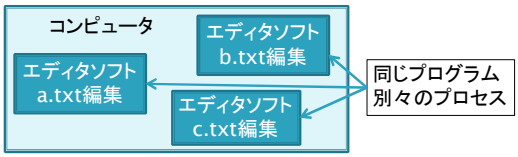
第7章 プロセスとその管理

花田　英輔
（このPowerPointは渡辺名誉教授作成のものを花田が一部改編した）

1

プログラム実行制御とプロセス(教科書7.1)

- プロセス＝プログラム実行の単位、タスクとも言う
- 一つのプログラムでも別々に動かせば別々に制御が必要⇒この制御単位がプロセス

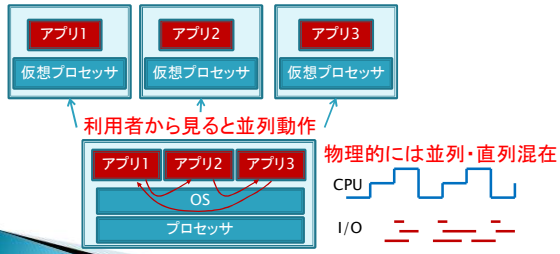


- 各実行単位に、プロセッサを占有しているように見える環境（仮想マシン環境）を提供
  ⇒仮想マシン提供の単位＝プロセス

2

プロセスによる並列処理

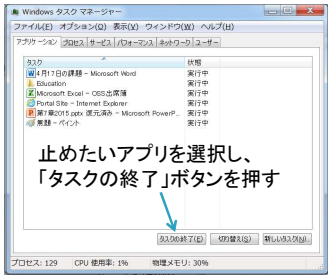
- プロセスは、利用者からは並列動作に見える
- 物理的には、入出力装置における処理等の並列部分とプロセッサにおける処理等の直列部分が混在



3

Windowsのプロセス管理

- Windowsでもプロセス管理は行われている



- Ctrl＋Alt＋Del（3つのキーの同時押し）で「タスクマネージャーの起動」を選択

やってみよう！

- あるプログラムが無反応になった場合、このウインドウで停止可能

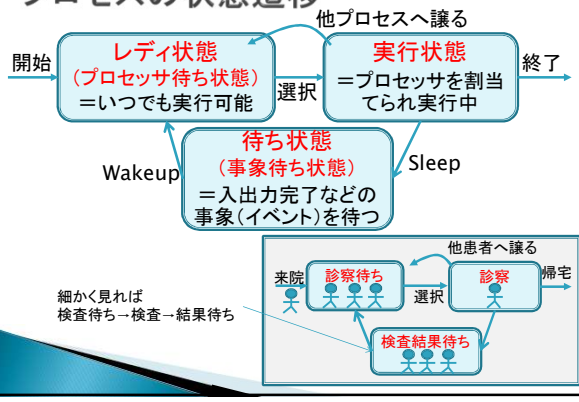
4

プロセスを構成するもの(教科書7.2)

- プロセッサの実行環境
  - 実行時＝プロセッサを割当てられ使用
  - 待ち時＝退避領域に実行環境保持（命令カウンタ、演算レジスタ、その他レジスタの内容）
- メモリ空間
- 開かれているファイル
- 親プロセスの情報
  - 当プロセスを生成したプロセスの情報
- 使用ユーザの情報
- これらの情報を保持するデータ構造＝プロセス記述子（プロセスの台帳、管理簿）

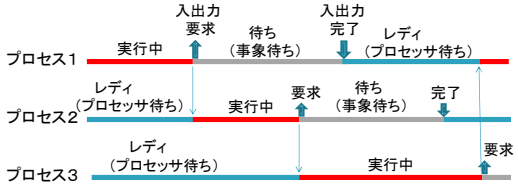
5

プロセスの状態遷移



6

プロセスの状態遷移の例

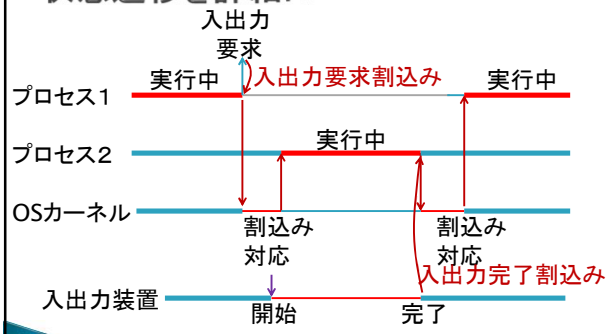


上記では、「入出力要求」によって切換え
「プロセッサを他プロセスへ譲る」切換えでは、待ち（事象待ち／イベント待ち）を経由せずレディ状態へ

これらの切換え時はOSが関与

7

状態遷移を詳細に



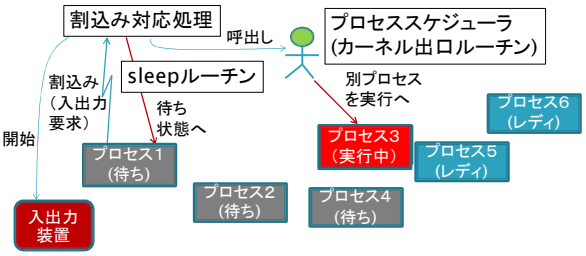
8

プロセススケジューラ(教科書7.3)

- プロセスの実行を制御するOSのプログラム
  - プロセスの状態管理
  - 次に実行すべきプロセスを選択
  - そのプロセスにプロセッサを渡す
- 割込み対応処理が終了後に呼出し、制御を渡すプロセスを決定

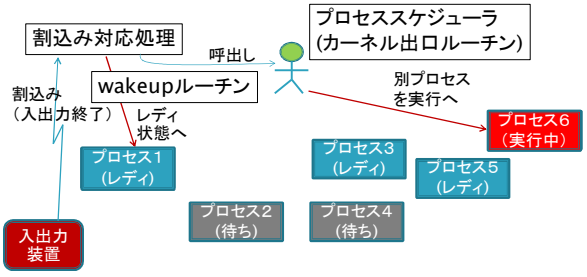
9

入出力要求割込み時点での動作



10

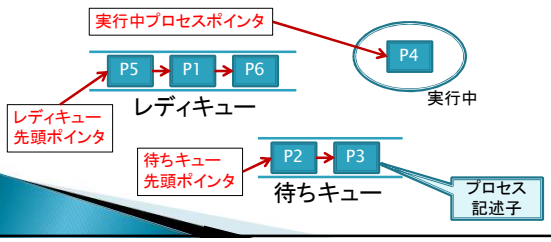
入出力完了割込み時点での動作



11

プロセススケジューラが使うデータ構造

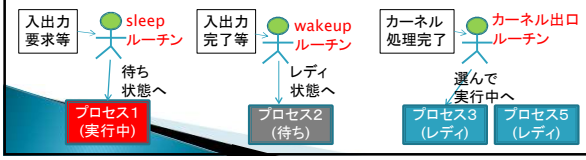
- 実行中プロセスポインタ＝実行中プロセスを指す
- レディキュー＝レディ状態のプロセスの行列
- 待ちキュー＝待ち状態のプロセスの行列
- プロセス記述子＝各プロセスを表現するデータ



12

プロセススケジューラのプログラム

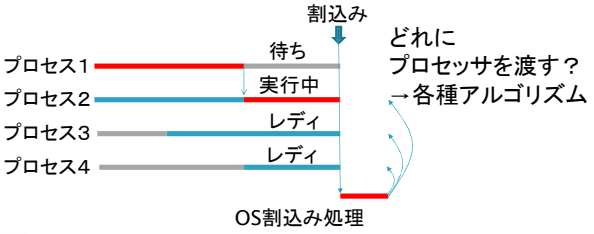
- sleepルーチン
  - 「実行中」プロセスを「待ち状態」に
  - 入出力要求時等
- wakeupルーチン
  - 指定した「待ち状態」プロセスを「レディ状態」に
  - 入出力完了時等
- カーネル出口ルーチン
  - 「レディ状態」プロセスの一つを選んで「実行中」に
  - カーネルの処理が終わり次に実行するプロセスが必要な時



13

プロセススケジューリングアルゴリズム (教科書7.4)

- 実行中および待ち状態にあるプロセスから、次に実行するプロセスをどう選ぶ？



14

各種スケジューリングアルゴリズム

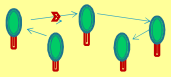
- 到着順(FCFS、First Come First Serve)
  - 先に来たものが先に実行
    - 単純。長時間実行プロセスがあると他が待たされる
- 実行時間最短のものから
  - プロセス実行完了までの平均時間が最小のものを先に
    - 実現困難(後の実行時間を知るのは困難)。前実行から推定
- 優先度順
  - あらかじめ与えた優先度の順に(仕事の重要度など)
    - 次々に到着⇒低優先度のプロセスはいつまでも処理なし

15

各種スケジューリングアルゴリズム(続)

- ラウンドロビン
  - 200ミリ秒程度の時間(タイムスライス)で順次切替え実行
  - 公平、切替えのオーバーヘッド大
- 優先度順で同一優先度のプロセス群はラウンドロビン
  - 上の組合せ。オンラインは優先度高・バッチは優先度低
- ダイナミックディスパッチング
  - 入出力が頻繁⇒優先度上げ。入出力がまれ⇒優先度下げ
  - 入出力が頻繁なら、渡しても直ぐに返してくれる
- 多段フィードバック
  - 最初は高い優先度。一定時間過ぎると優先度下げ
- いくつかの動作例の図示は教科書p.96～97に図示

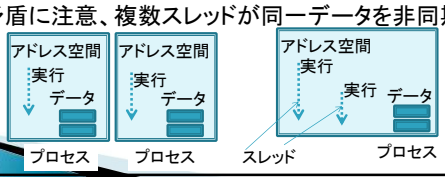
ラウンドロビン,round robin
(コマドリが木を渡り歩く習性から。諸説有)



16

スレッド(軽量プロセス)(教科書7.5)

- プロセス
  - 個別のアドレス空間を持つ
  - プロセス生成・データ交換のオーバーヘッド大
- スレッド(thread＝細線)
  - 1アドレス空間中に複数の実行制御単位を生成⇒スレッド
  - 生成の負荷小。データ交換容易
  - データ矛盾に注意、複数スレッドが同一データを非同期書換



17

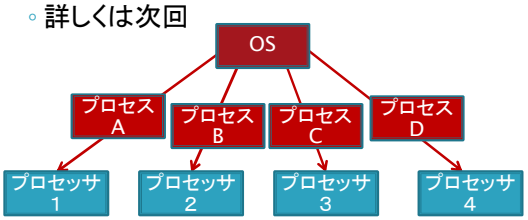
マルチプロセッサ(教科書7.6)

- 1台のコンピュータ上に複数のプロセッサ搭載
  - 非対称型マルチプロセッサシステムASMP
    - 数値演算等専用と汎用処理用のプロセッサ
  - 対称型マルチプロセッサシステムSMP
    - 全て汎用処理用、主流
- SMP:プロセス・スレッド単位で各プロセッサに割当て
  - OSが制御するので応用プログラム上は意識不要

18

マルチプロセッサの管理

- OSカーネルはプロセッサ間で排他的に処理
  - 親分は一人
  - 詳しくは次回



19

本日の課題

1. プロセスの3つの状態の間での遷移が発生する方向と、発生する理由を示せ。
2. (予習)プロセスの「排他制御」とは何か調べて記せ

- 今回のファイル名は“学籍番号-OS09.docx”
  (例:24238000-OS09.docx)としてください
- 締切:12月12日(金) 18:00 (遅れた場合は減点)

記載時の注意事項

- 参考資料(Webページ)がある場合は出典を書くこと
  - 出典を書かずに引用した場合は減点対象です

20