

第6回 デッドロック(2)

解決法1:デッドロックの防止

デッドロックの必要条件が成立しないようにする(何れか1つが否定できればよい)

必要条件	必要条件の成立を防止するための方法	評価
相互排除	全ての資源を共用できるようにする	実現不可能 ××
確保と待機	資源を確保したまま資源待ちしない ・必要な資源を実行開始時に全て要求 ・資源待ち時は、確保済み資源を解放	処理が複雑△ ・資源の使用率低下 ・再取得処理が必要
横取り不能	待機中プロセスから、資源を横取り (横取りした資源は、後で再割り当て)	制御が複雑× ・使用途中の資源の扱い
循環待機	資源型に順序を付け、アプリケーションには、 若番の資源型から順に要求するように プログラミングさせる	○ 他の条件を否定するのは困難(不可能)

デッドロックの防止:循環待機が成立しないようにする(資源の利用順序が1方向)  
(老番の資源を先に使用したい時は、若番の資源も先に確保してしまう)

問題点:APのプログラミングに制約が付く。資源の使用効率が非常に悪い。

デッドロック防止の例

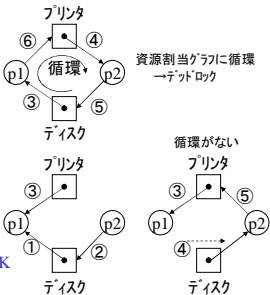
以下の順序で資源を要求するように決める

資源型	順序番号
磁気ディスク	1
プリンタ	2

- (1) P2が順番を守らない場合
- ①P1:磁気ディスク要求 OK
  - ②P2:プリンタ要求 OK
  - ③P2:磁気ディスク要求 待ち合わせ
  - ④P1:プリンタ要求 待ち合わせ

- (2) P2も順番を守る場合
- ①P1:磁気ディスク要求 OK
  - ②P2:磁気ディスク要求 待ち合わせ
  - ③P1:プリンタ要求 OK
  - ④P1:磁気ディスク開放 P2に割り当てOK
  - ⑤P2:プリンタ要求 待ち合わせ

プリンタを先に使う必要がある場合は？  
両方の資源を1度に要求  
(すぐに使わない資源を事前に確保する  
ので、使用効率が非常に悪い)



デッドロック防止(補足)

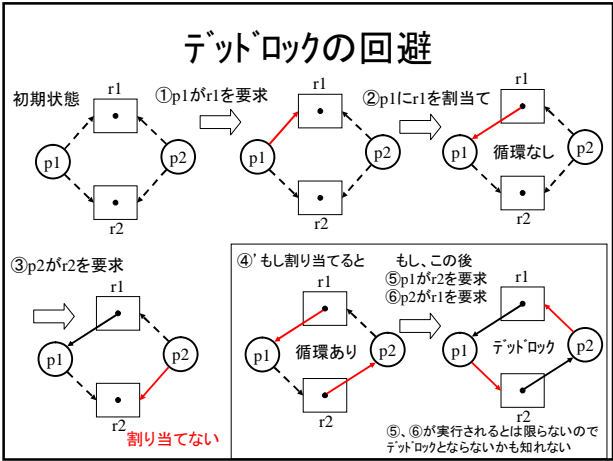
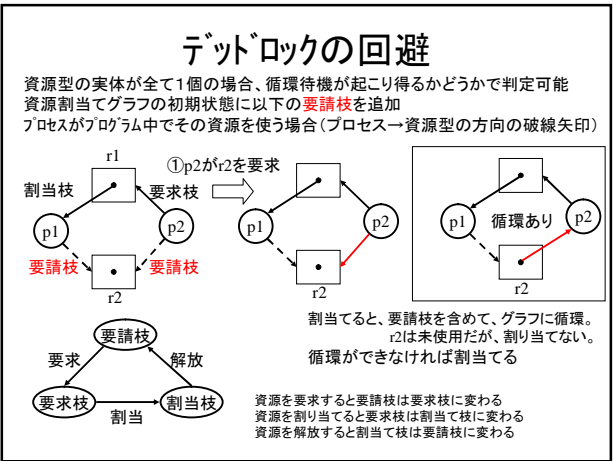
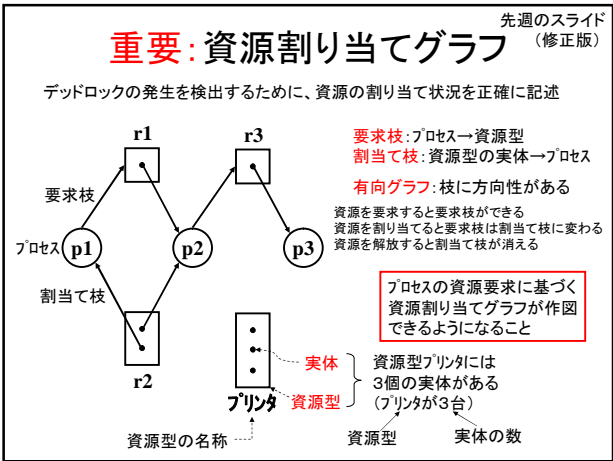
- 前スライドにおける決められた要求の順番
- ①磁気ディスク要求→②プリンタ要求
- P1:①磁気ディスク要求→③プリンタ要求→磁気ディスク解放→プリンタ解放
- (1) 順番を守らない場合
- P2(修正前):
  - ①磁気ディスク要求→②カードリダ要求→プリンタ解放→磁気ディスク解放
- (2) 順番を守るように変更
- P2(修正後):
  - ①磁気ディスク要求→②プリンタ要求→プリンタ解放→磁気ディスク解放
- 要求の順番を合わせると、循環待機が発生しなくなる

解決法2:デッドロックの回避

- プロセスが資源を要求した時に、その資源を割り当てるとデッドロックが起り得るかどうかを判定。起り得る場合は、資源が空いていても割り当てない。
- 判定アルゴリズム(銀行家アルゴリズム)の考え方
  - 資源の割り当て状態を管理(各プロセスへの割り当て数、空き数)
  - 各プロセスは要求する資源の最大数を申告。
  - 資源を割り当てても安全な順序(空き資源を使い全プロセスが終了できる順序)が存在するかどうかを判定
    - 存在:デッドロックが起らないので資源を割り当てる
    - 存在しない:要求を待たせる
- 問題点
  - 処理負荷が大:要求の都度、判定アルゴリズムの実行が必要
  - 資源の最大数の宣言が必要(予測できない場合もある)
  - 資源の使用効率が悪い

銀行家アルゴリズム(資源型が1種類の場合)

- プロセスPiへの割り当て済み資源数Ai、Piの最大使用資源数Mi
- システム全体の残り資源数:W
- 今、プロセスPjが資源をRj個要求した。資源を割り当てて良いか？
- W<Rjの場合(資源が足りない)
  - Pjの要求を待たせる。
- W≥Rjの場合(資源は足りている)
  - Aj=Aj+Rj, W=W-Rj(Pjに仮割り当て)。安全な順序を探す。
    - (Mi-Ai)≤Wが成立するプロセスPiを探す
    - Fi=終了可能
    - W=W+Ai
  - 安全な順序あり:全プロセスが、Fi=終了可能の場合
    - Pjが要求した資源を割り当て。
  - 安全な順序が無い:Fi≠終了可能 というプロセスがある場合
    - 変数を元に戻し、Pjの要求を待たせる



### 解決法3: デッドロックの検出と回復

- プロセスが資源を要求時、資源に空きがあれば無条件に割当てる
- デッドロック検出アルゴリズムを定期的に起動
- 検出アルゴリズムの考え方
  - 資源の使用状況を管理 (各プロセスへの割当て数、割り当て待ち数、システム全体の空き資源数)
  - 空きの資源を使い、全プロセスの割り当て待ち資源を満足する順序が存在するかどうかを判定する。
    - 現在、割当て待ち資源を与えられれば、プロセスは終了できる可能性がある。(後で、デッドロックになるかも知れないが、今は心配しない)
    - 要求を満足するプロセスへの割当て資源は、空き数として加算 (要求が満たされれば終了でき、確保中の資源を返却できる可能性がある)
  - 上記の順序が存在しなければ、デッドロック
- デッドロック検出時
  - 該当プロセスをロールバック (強制終了し、チェックポイントから再開)
  - または 確保済み資源を横取り

### 検出アルゴリズム (資源型が1種類の場合)

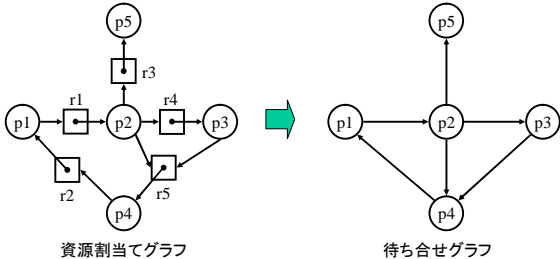
- プロセス $P_i$ への割り当て済み資源数:  $A_i$ 、 $P_i$ の割り当て待ち資源数 $R_i$
- システム全体の残り資源数:  $W$
- $A_i = 0$ の $P_i$ について、 $F_i = \text{非デッドロック}$  (注1)
- $A_i > 0$ のプロセスについて、要求が満たせる順序を探す。
  - $R_i \leq W$ が成立するプロセス $P_i$ を探す (注2)。
  - $F_i = \text{非デッドロック}$
  - $W = W + R_i$  (注3) } 繰り返し
- 全プロセスが、 $F_i = \text{非デッドロック}$ ならば、デッドロックは無い。
- $F_i \neq \text{非デッドロック}$ のプロセスはデッドロック。
- 注1: デッドロックの必要条件 (確保と待機) を満足しない。
- 注2: 残りの資源で要求が満足できればデッドロックではない。
- 注3: 要求が満たされるプロセスは終了でき、確保中の資源を返却できる可能性がある。

### 検出アルゴリズム (補足)

- 総資源数8、残り資源数0
- P1: 資源を2個確保。4個要求 (待ち合せ) → ④非デッドロック
- P2: 資源を2個確保 → ②非デッドロック
- P3: 資源を2個確保。2個要求 (待ち合せ) → ③非デッドロック
- P4: 資源確保なし。 → ①非デッドロック
- P2が終了して返した資源で、P3の要求が満たされ、P3が終了して返した資源でP4の要求も満たされる
- 総資源数8、残り資源数1
- P1: 資源を2個確保。4個要求 (待ち合せ) → デッドロック
- P2: 資源を2個確保 → ②非デッドロック
- P3: 資源を3個確保。4個要求 (待ち合せ) → デッドロック
- P4: 資源確保なし。 → ①非デッドロック
- P2が終了して返した資源では、P1、P3の要求を満たさない

重要: 待ち合わせグラフ

実体の数が1個の場合のデッドロック検出アルゴリズム  
資源割当てグラフ: 循環があればデッドロック  
(循環の検出さえできれば良い)  
待ち合わせグラフ: プロセス間の待ちの関係を示す(資源割当てグラフを簡略化)  
待ち合わせグラフに循環があればデッドロック



検出と回復における考慮要因

- 検出アルゴリズムの呼び出し契機
  - デッドロックの頻度
  - デッドロック発生時に影響を受けるプロセス数
- デッドロックからの回復方法
  - プロセスの終了(その結果資源も横取り)
    - デッドロック中の全プロセス
    - 循環がなくなるまで、順番に終了(誰を選ぶか)
    - ファイル書き込み中の対処、プリンタの再設定等も必要
  - 資源の横取り
    - 犠牲者の選択(優先度、どこまで処理したか、影響度)
    - ロールバック(どこまで戻すか、一時退避、チェックポイント)
    - 飢餓状態(犠牲者の偏り、経費要因とロールバック回数)

デッドロック処理技法(まとめ)

名称	考え方	処理方法	処理 負荷	利用 効率	利便 性
防止 (静的)	循環待機の条件成立を防止	資源型に順序番号。各プロセスは、若番の資源型から順に要求。	○	×	×
回避 (動的)	資源割り当てを制限しデッドロックの可能性を回避	資源要求の度に、銀行家アルゴリズムで安全な順序の存在をチェック。 存在しなければ資源を割当てない	×	△	△
検出と回復	資源割り当てを制限せず、デッドロック発生を許容	検出アルゴリズムを定期的に実行 デッドロックのプロセスをロールバック or 資源の横取り	△	○	○
放置	何もしない (ユーザまかせ)	固まったらマニュアルで再開 (Unix、Windows含むいくつかのOS)	○	○	×

注1: プロセスが資源を要求する度に銀行家アルゴリズムを実行する必要がある  
注2: 使用しない資源を長時間確保することがある(すぐ使わなくても要求が必要)  
注3: デッドロックになるとは限らない場合でも、資源の割り当てを制限する  
注4: 資源の利用順序を守るようにプログラミングする必要がある  
注5: 使用する資源の最大数を宣言する必要があるが、予測が困難な場合がある

8.7 複合的アプローチ

内部資源: OSが使用する資源(PCBなど)  
主記憶: 利用者のジョブが使用する主記憶  
ジョブ資源: 割り当てできる装置、ファイル  
スワップ領域: 2次記憶上の各ジョブの領域

上記資源の性質に応じてデッドロック処理方法を選択