デッドロック

デッドロックとは

- デッドロックとは
  待ちが引き起こす「待合い」(３すくみ)
  のこと

デッドロックとは

- デッドロックとは
  待ちが引き起こす「待合い」(３すくみ)のこと
- ３すくみ状態だと、
  いくら待っていても、先へ進めない

デッドロックとは

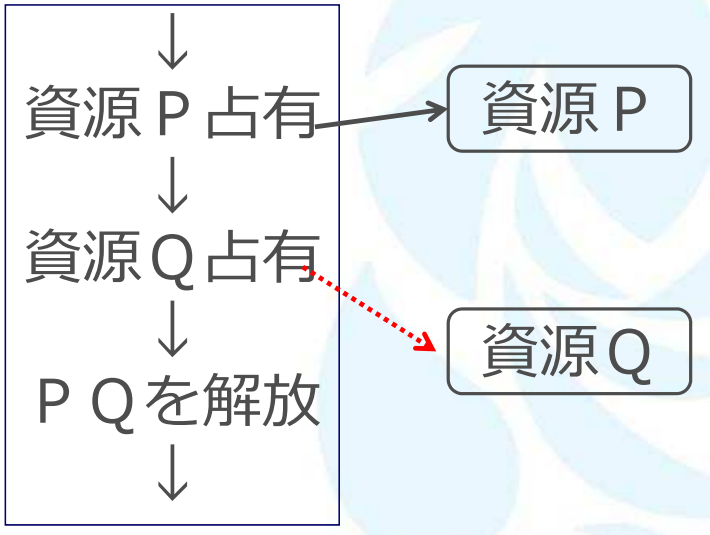
- デッドロックとは
  待ちが引き起こす「待合い」(３すくみ)のこと
- ３すくみ状態だと、
  いくら待っていても、先へ進めない
- イメージ：
  ＡがＢの終了を待ち、ＢがＡの終了を待つ

排他制御でのデッドロック

- 排他制御（資源待ち）の場合では、
  Ａが資源Ｐを確保した後、資源Ｑ待ち
  Ｂが資源Ｑを確保した後、資源Ｐ待ち

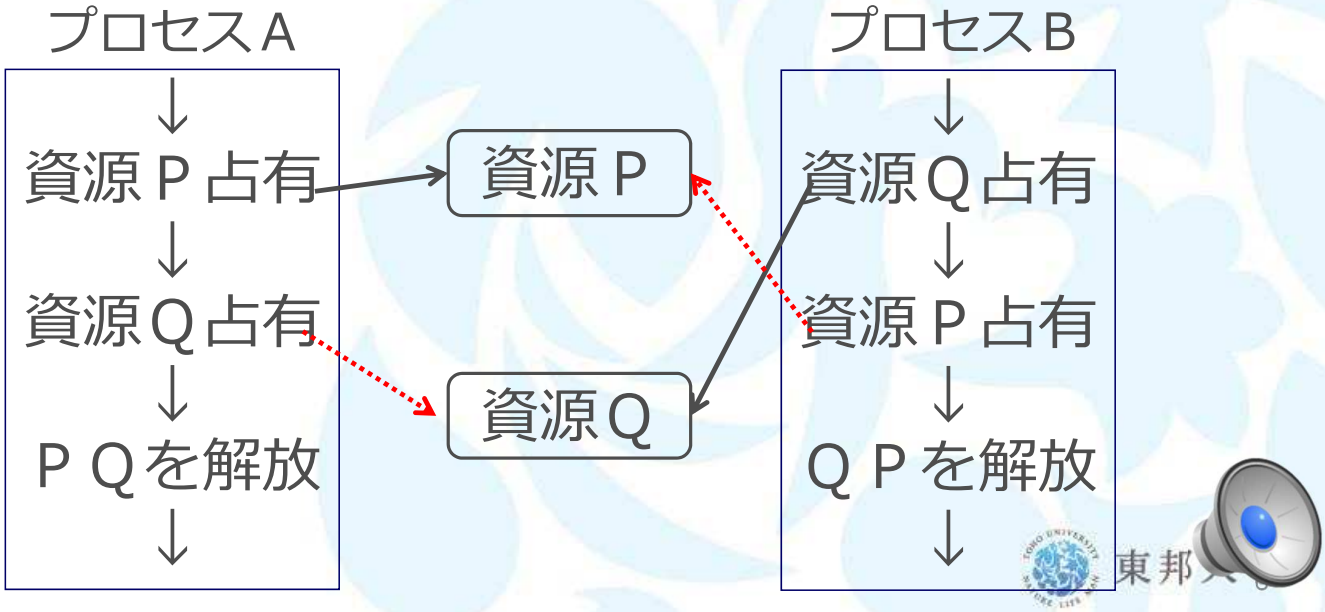
排他制御でのデッドロック

- 排他制御（資源待ち）の場合では、
  Ａが資源Ｐを確保した後、資源Ｑ待ち
  Ｂが資源Ｑを確保した後、資源Ｐ待ち



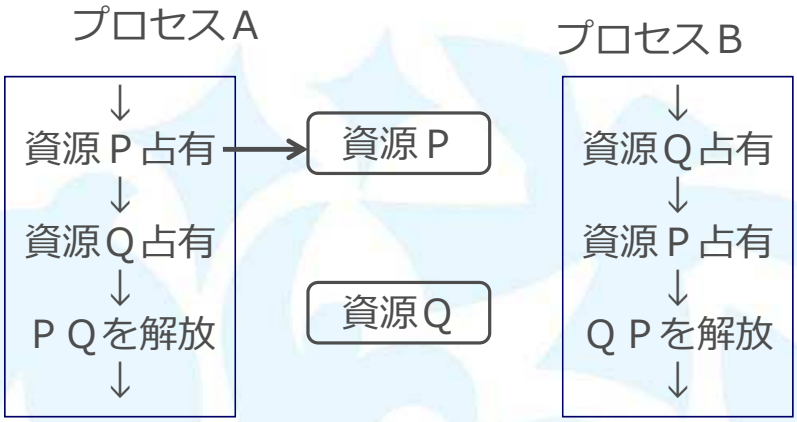
排他制御でのデッドロック

- 排他制御（資源待ち）の場合では、
  Ａが資源Ｐを確保した後、資源Ｑ待ち
  Ｂが資源Ｑを確保した後、資源Ｐ待ち



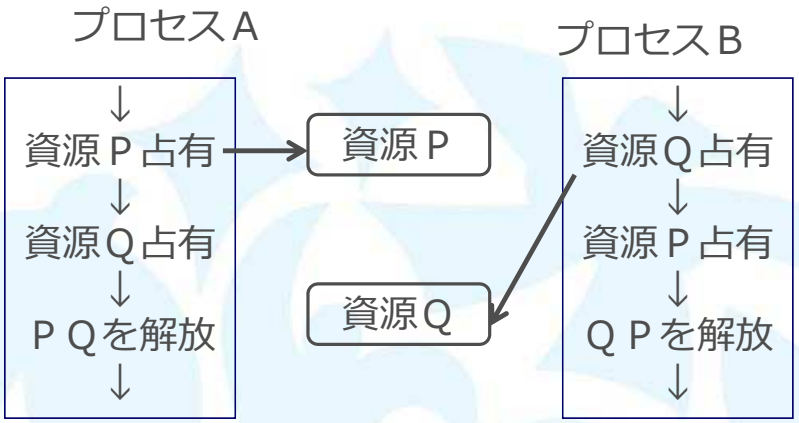
排他制御でのデッドロック

- Ａが Ｐ を確保した後、



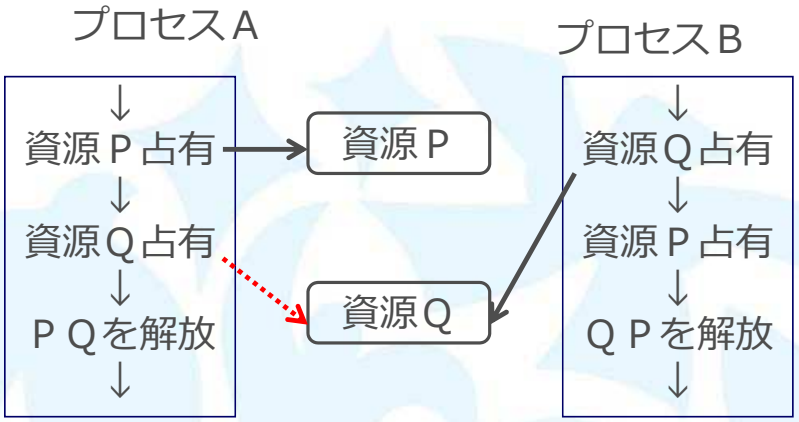
排他制御でのデッドロック

- ＡがＰを確保した後、
  Ｑをとる前に
  　Ｂが Ｑを確保した。



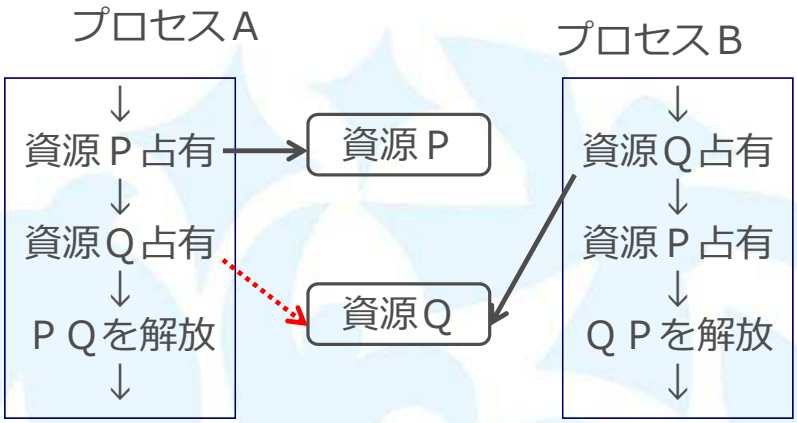
排他制御でのデッドロック

- ＡがＰを確保した後、
  Ｑをとる前に
  　Ｂが Ｑを確保した。
  Ａが Ｑを取ろうとしても
  　Ｂが既に占有している



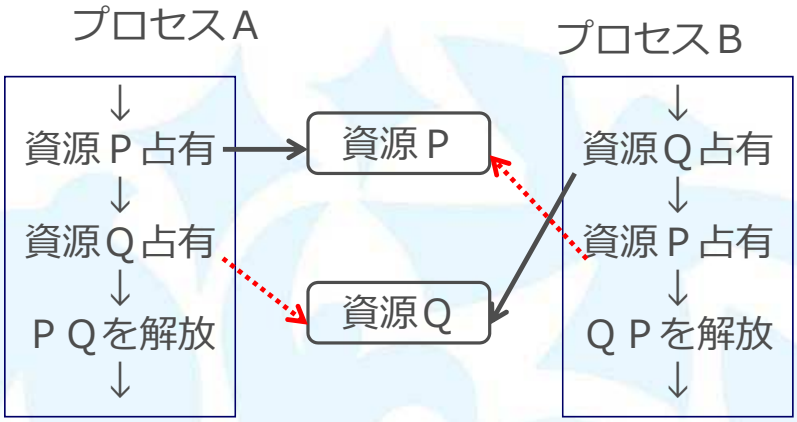
排他制御でのデッドロック

- ＡがＰを確保した後、
  Ｑをとる前に
  　Ｂが Ｑを確保した。
  Ａが Ｑを取ろうとしても
  　Ｂが既に占有している
- ＢはＱを確保した後、



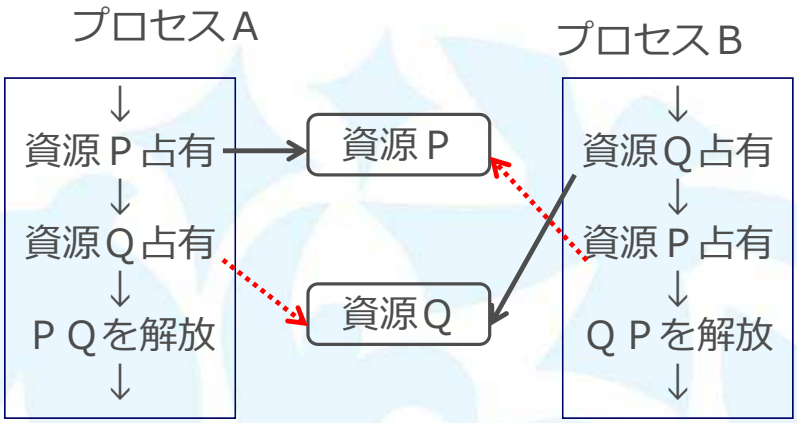
排他制御でのデッドロック

- ＡがＰを確保した後、
  Ｑをとる前に
  　Ｂが Ｑを確保した。
  Ａが Ｑを取ろうとしても
  　Ｂが既に占有している
- ＢはＱを確保した後、
  Ｐを取りたいが
  　Ａが既に占有している



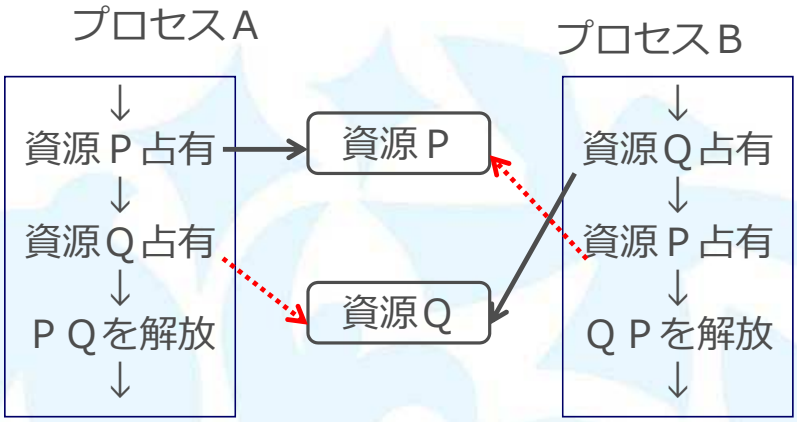
排他制御でのデッドロック

- ＡがＰを確保した後、
  Ｑをとる前に
  　ＢがＱを確保した。
  ＡがＱを取ろうとしても
  　Ｂが既に占有している
- ＢはＱを確保した後、
  Ｐを取りたいが
  　Ａが既に占有している
- 結果は
  ＡはＰを確保したまま
  　Ｑの空くのを待つ
  ＢはＱを確保したまま
  　Ｐの空くのを待つ



排他制御でのデッドロック

- ＡがＰを確保した後、
  Ｑをとる前に
  　ＢがＱを確保した。
  ＡがＱを取ろうとしても
  　Ｂが既に占有している
- ＢはＱを確保した後、
  Ｐを取りたいが
  　Ａが既に占有している
- 結果は
  ＡはＰを確保したまま
  　Ｑの空くのを待つ
  ＢはＱを確保したまま
  　Ｐの空くのを待つ



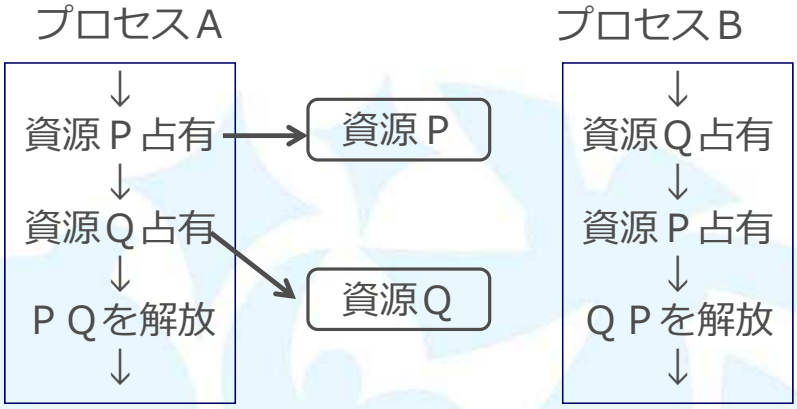
Ａ，Ｂどちらも正常実行だが
資源の空き待ちで、
先へ進めない状態になって

原因の分析

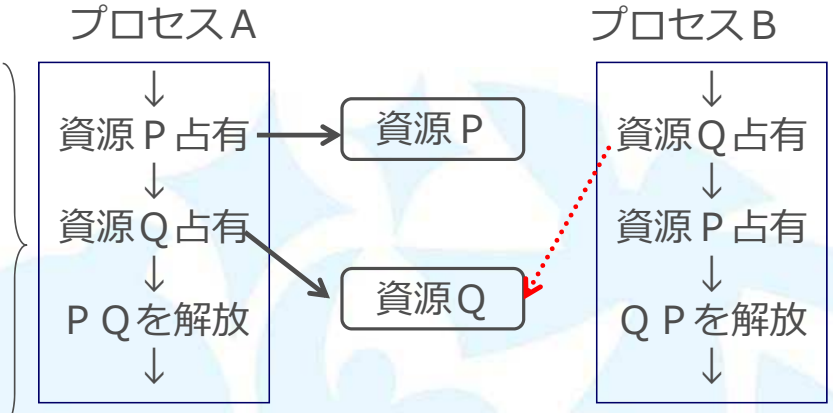
- うまく行くこともある

原因の分析

- うまく行くこともある
- AがPを確保した後、続けてQを確保できた
  この時BはまだQを確保していない

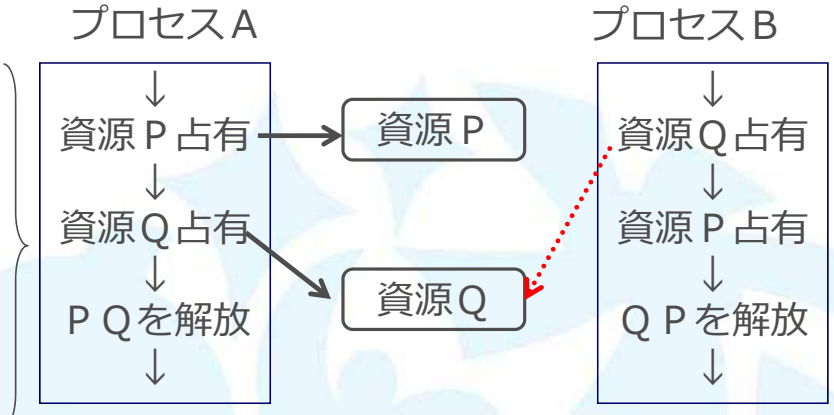


原因の分析



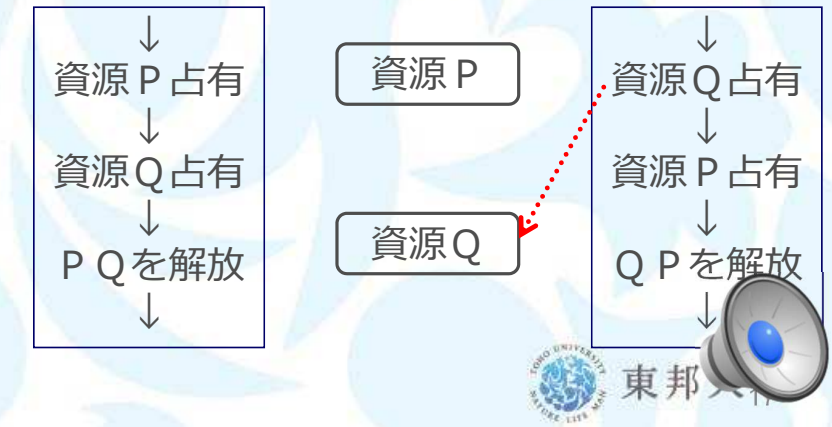
- うまく行くこともある
- AがPを確保した後、続けてQを確保できた
  この時BはまだQを確保していない
- Bはその後Qを確保しようとするが、既にAに取られているので待つ

原因の分析



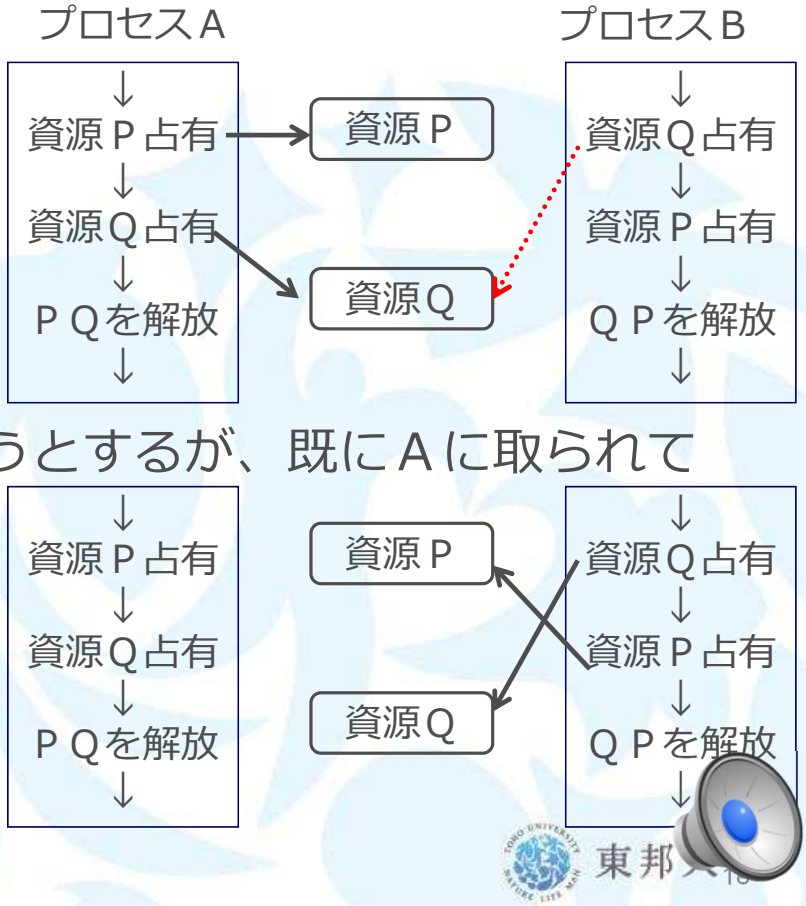
- うまく行くこともある
- AがPを確保した後、続けてQを確保できた
  この時BはまだQを確保していない
- Bはその後Qを確保しようとするが、既にAに取られているので待つ

- Aは続けて処理・終了し、PとQを解放する



原因の分析

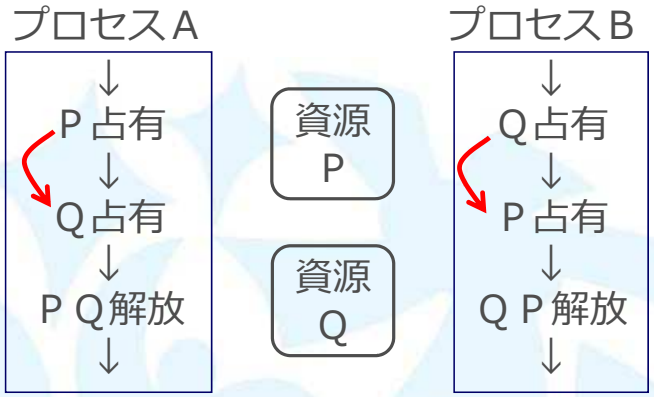
- うまく行くこともある
- ＡがＰを確保した後、続けてＱを確保できた
  この時Ｂはまだ Ｑを確保していない
- Ｂはその後Ｑを確保しようとするが、既にＡに取られているので待つ
- Ａは続けて処理・終了し、ＰとＱを解放する
- ＢはＱが空いたので Ｑを確保、続けてＰ確保 そのまま処理・終了する



原因の分析

整理すると

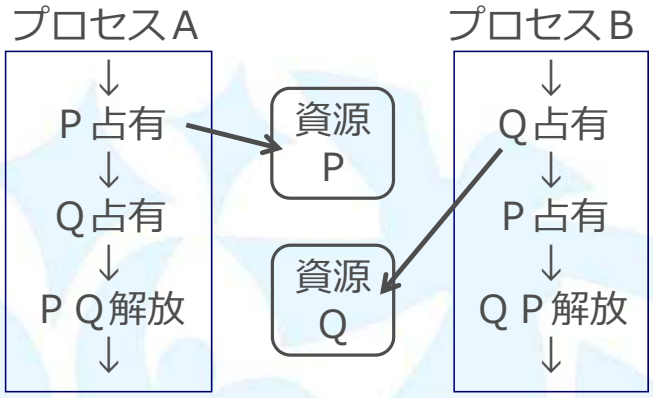
- プロセスＡとＢが、２つの資源ＰとＱを 逆の順序で確保したい



原因の分析

整理すると

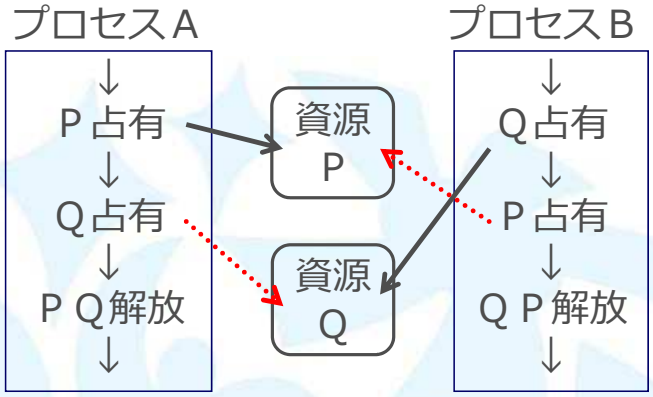
- プロセスＡとＢが、
  ２つの資源ＰとＱを
  逆の順序で確保したい
- ちょうど１つずつ、
  ＡがＰを、ＢがＱを
  確保した状態になると



原因の分析

整理すると

- プロセスＡとＢが、
  ２つの資源ＰとＱを
  逆の順序で確保したい
- ちょうど１つずつ、
  ＡがＰを、ＢがＱを
  確保した状態になると
- ＡはＱを、ＢはＰを待つよう
  になってデッドロックになる



デッドロックが発生する (必要十分) 条件

もう少し一般的にすると（教科書）４つの条件

- 相互排他（排除)があって
- 資源確保のとき、取れなければ待つ方式で
  （取れなければすぐあきらめて、今確保した
  資源も全部解放する方式ではなく）
- 資源の横取り不可で
  （現在の占有者から資源を横取りできない）
- 循環待ちであるとき
  （待ちがお互いにぐるっと循環している）

デッドロックが発生する (必要十分) 条件

もう少し一般的にすると（教科書）４つの条件

- 相互排他（排除)があって
- 資源確保のとき、取れなければ待つ方式で
  （取れなければすぐあきらめて、今確保した
  資源も全部解放する方式ではなく）
- 資源の横取り不可で
  （現在の占有者から資源を横取りできない）
- 循環待ちであるとき
  （待ちがお互いにぐるっと循環している）

必要十分条件 ⇒ どれか１つでも不成立なら発生し

資源割付グラフでの循環

- 循環待ちは
  資源割付グラフを描くとわかる
  （資源割付グラフ上で循環しているとＮＧ）

資源割付グラフでの循環

- 循環待ちは
  資源割付グラフを描くとわかる
  （資源割付グラフ上で循環しているとＮＧ）
- 資源割付グラフは、
  プロセス（楕円）と資源（四角）をノードとし
  ２種類のエッジを書く
    実線矢印＝プロセスが資源を要求している
    破線矢印＝資源がプロセスに割当済みである
  この上でサイクルが存在するとＮＧ

ダイニング・フィロソファー問題

- Dining Philosophers
- デッドロックの（計算機科学者間で）有名な例
- Ｎ人の哲学者が食事
  フォークが２本必要
  だが１本ずつしかない

全員が片方ずつフォークを取ると、食べられない

デッドロックのまとめ

- デッドロックとは

デッドロックのまとめ

- デッドロックとは
  資源の利用待ちで待ち状態に入り、
  お互いの待ち解除条件を妨害していて
  ３すくみ状態にあること

デッドロックのまとめ

- デッドロックとは
- デッドロックの起こる（必要十分）条件は

デッドロックのまとめ

- デッドロックとは
- デッドロックの起こる（必要十分）条件は
  - 相互排他（排除)があって
  - 資源確保のとき、取れなければ待つ方式で
  - 資源の横取り不可で
  - 循環待ちであるとき

  １つでも条件が外れればデッドロックしない

デッドロックのまとめ

- デッドロックとは
- デッドロックの起こる（必要十分）条件は
- 資源割付グラフで、循環待ちがあるかがわかる

デッドロックが何かが理解できましたか？

次へ