
# SISTEMAS OPERATIVOS

## Interblocagem - deadlocks

António Godinho

# INTERBLOCAGEM

Interblocagem (deadlock):Um conjunto de processos bloqueados, cada um deles detendo um recurso e à espera de outro recurso que está na posse de um dos processos do conjunto.

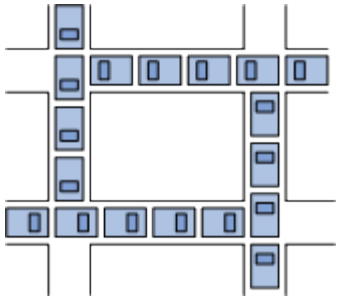
Exemplo 1: Um sistema tem duas unidades de bandas magnéticas. Dois processos, cada um detendo o acesso a uma das unidades e à espera da segunda.

Exemplo 2: dois semáforos A e B, inicializados a 1. P0 aguarda por semáforo B que nunca será assinalado pelo facto do processo P1 estar bloqueado à espera que o semáforo A seja assinalado.

```
/* processo P0 */
...
espera(&A);
espera(&B);
...
assinala(&A);
assinala(&B);
```

```
/* processo P1 */
...
espera(&B);
espera(&A);
...
assinala(&B);
assinala(&A);
```

Exemplo 3:cruzamento.

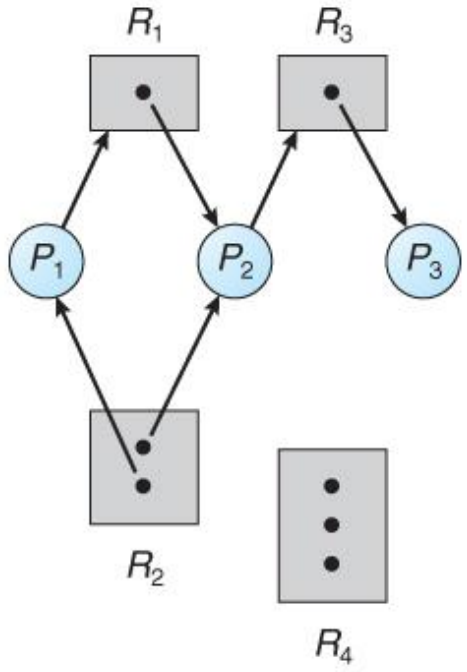


# CONDIÇÕES (DE COFFMAN) PARA O DEADLOCK

1. Exclusão mútua: apenas um processo pode, em determinado instante, aceder ao recurso.
2. Posse e espera: um processo detém pelos menos um recurso e encontra-se à espera de adquirir recursos adicionais na posse de outros processos.
3. Não preempção: um recurso apenas é libertado de forma voluntária pelo processo que o detém.
4. Espera circular: Um conjunto de processos bloqueados {P0, P1, …, Pn} de tal modo que P0 está à espera de um recurso alocado a P1, P1 está à espera de um recurso alocado a P2, P3…, Pn-1 está à espera de um recurso alocado a Pn e Pn está à espera de um recurso alocado a P0.

# GRAFO DE ALOCAÇÃO DE RECURSOS (1)

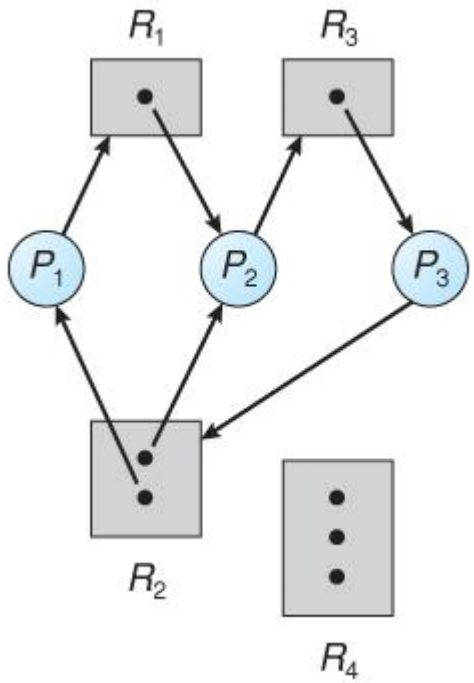
- A interblocagem entre processos pode ser descrita através de um grafo orientado (Modelação proposta por Holt).
  - Os vértices deste grafo representam o conjunto de processos (P1..PN) e dos recursos (R1..RN).
  - Um arco orientado de Pi para Rj, designado Pi-->Pj (pedido), significa que o processo Pi pretende aceder a uma instância do recurso Rj.
  - Um arco orientado de Rj para Pi, designado por Rj-->Pi (alocado), significa que uma instância do recurso Rj está alocada ao processo Pi.
  - Exemplo: R2 -->P1 ; P1-->R1 ; R1-->P2 ; R2-->P2 ; P2-->R3 ; R3-->P3. O processo P1 detém uma instância do recurso R2 e pretende aceder ao recurso R1. O processo P2 detém o recurso R1, uma instância de R2 e pretende aceder ao recurso R3. O processo P3 detém o recurso R3.



# GRAFO DE ALOCAÇÃO DE RECURSOS (2)

- Num grafo sem ciclos, não existe interblocagem.
- Num grafo com ciclos, a interblocagem pode existir. Se cada recurso tem apenas uma instância e existe um ciclo, então existe interblocagem. Se existirem várias instâncias para os recursos, poderá existir ou não interblocagem.
- A existência de um ciclo é necessária, mas não é suficiente para existir interblocagem.
- Exemplo de grafo com ciclos e interblocagem:

Ciclo 1: P1-->R1-->P2-->R3-->P3-->R2-->P1
Ciclo 2: P2-->R3-->P3-->R2-->P2
Interblocagem entre os processos P1, P2 e P3: P2 aguarda por R3 que está alocado a P3, P3 aguarda por uma instância de R2 que pode ser libertado por P1 ou P2, P1 aguarda por R1 que está alocado a P2.

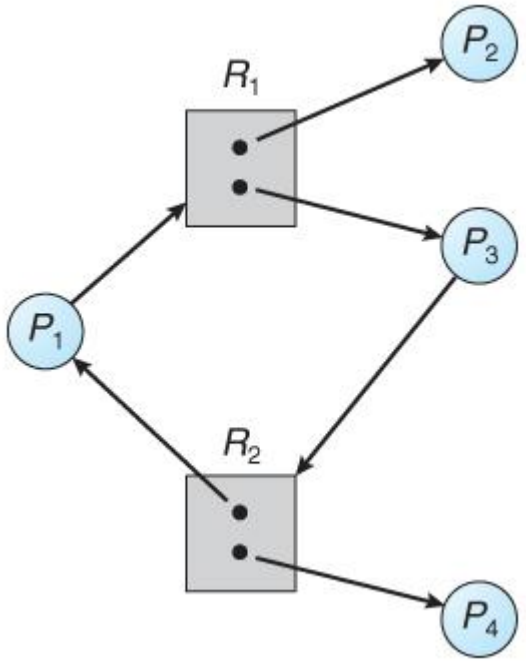


# GRAFO DE ALOCAÇÃO DE RECURSOS (2)

- Exemplo de grafo com ciclo mas sem interblocagem:

- Ciclo: P1-->R1-->P3-->R2-->P1

- Não existe interblocagem dado que P4 pode libertar uma instância de R2 e assim quebrar o ciclo.



# INTERBLOCAGEM: SOLUÇÕES

O problema da interblocagem pode ser tratado:

- por prevenção ou evitação, recorrendo a um protocolo que previna ou evite o estado de interblocagem;
- por detecção e recuperação, permitindo que o sistema entre num estado de interblocagem, que o detecte e que recupere desse estado;
- ignorando o problema (algoritmo da avestruz) (utilizado por Windows e Unix – considera-se que deadlocks ocorrem muito raramente – relação custo/benefício).

A última opção é a normalmente considerada pela maioria dos sistemas operativos. É deixada ao programador de aplicações a responsabilidade de tratar do problema da interblocagem.

# INTERBLOCAGEM: PREVENÇÃO

Para que ocorra a interblocagem, devem ser verificadas as quatro condições referidas anteriormente (exclusão mútua, posse e espera, não preempção, espera circular). Se um sistema não permitir qualquer uma destas quatro condições, previne-se a interblocagem.

Estratégias para prevenir a interblocagem (propostas por Havender):

- Cada processo deve, de uma só vez, solicitar todos os recursos necessários e não prosseguirá sem que estes lhe tenham sido alocados (impedir posse e espera). Normalmente, isto conduz a uma alocação ineficiente de recursos, reduzindo, também, o grau de multiprogramação.
- Se a um processo lhe for negado o acesso a um novo recurso, este deverá libertar os recursos que detém e, se necessário, voltar a solicitá-los novamente (preempção). De notar que nem todos os recursos podem ser sujeitos à preempção sem que exista perda de dados ou de trabalho já realizado.
- Impor o acesso aos recursos por uma ordem linear crescente (impedir espera circular). A cada recurso é atribuído um número de ordem. Cada processo deve solicitar os recursos por ordem crescente.

# INTERBLOCAGEM: EVITAÇÃO (1)

Algoritmo do banqueiro (proposto por Dijkstra): um banqueiro garante ou não crédito (recursos) aos clientes (processos); Nem todos os clientes (processos) precisam de toda a linha de crédito (recursos) disponível para eles.

- Permite que um sistema evite a ocorrência de interblocagem através de um adequado controlo na afectação de recursos aos processos.
- Impõe menos restrições do que as estratégias de prevenção, procurando melhorar a utilização de recursos.
- Os recursos são agrupados em tipos de modo a que dois recursos do mesmo tipo oferecem a mesma funcionalidade.

# INTERBLOCAGEM: EVITAÇÃO (1)

Algoritmo do banqueiro (proposto por Dijkstra) - Características do algoritmo:

- O sistema partilha um número fixo de recursos por entre um número fixo de processos.
- Cada processo deve, à priori, estabelecer o número máximo de recursos que serão necessários para terminar a tarefa.
- O sistema aceita um pedido de um processo se este não exceder o número total de recursos.
- Um processo pode ficar à espera de recursos adicionais, mas o sistema garante uma espera limitada no acesso aos recursos.
- Quando um processo obtém todos os recursos necessários, estes são utilizados e libertados num intervalo de tempo limitado.

# INTERBLOCAGEM: EVITAÇÃO (2)

Estado seguro de alocação de recursos: o sistema garante que todos os processos podem completar as suas tarefas. No exemplo seguinte, o número de recursos disponíveis é T-(1+4+5)=12-10=2. Neste caso, o sistema encontra-se num estado seguro, sem possibilidade de interblocagem, pois o processo P2 pode obter os 2 últimos recursos necessários para completar a sua tarefa, libertando, em seguida, os 6 recursos utilizados que permitirão aos processos P1 e P3 terminarem a sua tarefa.

N: Número de processos (=3)
T: Número de recursos (=12)

| Processo | Max(i) | Aloc(i) | EmFalta(i) |
|---|---|---|---|
| P1 | 4 | 1 | 3 |
| P2 | 6 | 4 | 2 |
| P3 | 8 | 5 | 3 |

Max(i) : Número máximo de recursos necessários ao processo Pi
Aloc(i): Número de recursos alocados ao processo Pi
EmFalta(i): Número de recursos necessários para o processo Pi completar a sua tarefa = Max(i)-Aloc(i)

# INTERBLOCAGEM: EVITAÇÃO (2)

Estado não seguro de alocação de recursos: No exemplo seguinte, o número de recursos disponíveis é T-(8+2+1)=12-11=1. Neste caso, estando apenas disponível 1 recurso, pode existir a interblocagem. Por exemplo, se P1 obtiver acesso ao último recurso, nenhum dos processos poderá completar a sua tarefa.

| Processo | Max(i) | Aloc(i) | EmFalta(i) |
|---|---|---|---|
| P1 | 10 | 8 | 2 |
| P2 | 5 | 2 | 3 |
| P3 | 3 | 1 | 2 |

Nota: o facto de o sistema se encontrar num estado não seguro, não implica a ocorrência de interblocagem. O que implica é a existência de uma sequência de alocação de recursos que conduz à interblocagem.

# INTERBLOCAGEM: EVITAÇÃO (3)

Transição de estado seguro para estado não seguro:

N: 3
T: 12
Rec. Disponíveis:2 (estado seguro)

| Processo | Max(i) | Aloc(i) | EmFalta(i) |
|---|---|---|---|
| P1 | 4 | 1 | 3 |
| P2 | 6 | 4 | 2 |
| P3 | 8 | 5 | 3 |

P3 solicita um recurso
Rec. Disponíveis: 1
Estado não seguro

| Aloc(i) | EmFalta(i) |
|---|---|
| 1 | 3 |
| 4 | 2 |
| 6 | 2 |

Princípio adoptado no algoritmo do banqueiro de Dijkstra: Permitir uma nova alocação de recursos desde que se mantenha num estado seguro. No exemplo anterior, a solicitação de P3 não seria autorizada, não lhe sendo alocado um novo recurso. P3 aguardará pela disponibilidade desse recurso.

Algoritmo para verificar se o estado é seguro:

```
1:   W=RecursosDisponíveis
     Terminou[i] = false para i = 1..N
2:   Encontrar i (1..N) tal que Terminou[i]==false e EmFalta[i]<=W
3:   Se existir i então
          Terminou[i]=true
          W=W+Aloc[i]
          Ir para 2:
4:   Se Terminou[i]==true para i=1..N
       então estado seguro,
       senão estado não seguro
```

# INTERBLOCAGEM: EVITAÇÃO (4)

Algoritmo do banqueiro (exemplo 1):

| Processo | Max(i) | Aloc(i) | EmFalta(i) |
|---|---|---|---|
| P1 | 4 | 1 | 3 |
| P2 | 6 | 4 | 2 |
| P3 | 8 | 5 | 3 |

N: Número de processos (=3)

T: Número de recursos (=12)

1: W=12-(5+4-1)=2
Terminou[i] = false para i = 1..N
2: i=2 (Terminou[2]==false e EmFalta[2]<=W)
3: Terminou[2]=true
W=W+Aloc[2] =2+4=6
2: i=1 (Terminou[1]==false e EmFalta[1]<=W)
3: Terminou[1]=true
W=W+Aloc[2] =6+1=7
2: i=3 (Terminou[3]==false e EmFalta[3]<=W)
3: Terminou[3]=true
W=W+Aloc[3] =7+5
4: Terminou[i]==true para i=1..N então estado seguro

Algoritmo do banqueiro (exemplo 2):

| Processo | Max(i) | Aloc(i) | EmFalta(i) |
|---|---|---|---|
| P1 | 5 | 1 | 4 |
| P2 | 3 | 1 | 2 |
| P3 | 9 | 5 | 4 |

N: Número de processos (=3)

T: Número de recursos (=9)

1: W=9-(1+1+5)=2
Terminou[i] = false para i = 1..N
2: i=2 (Terminou[2]==false e EmFalta[2]<=W)
3: Terminou[2]=true
W=W+Aloc[2] =2+1=3
2: Não existe i
4: Terminou[1]==false e Terminou[3]==false
então estado não seguro

# INTERBLOCAGEM: DETECÇÃO E REPARAÇÃO

Detecção:

- Usado em sistemas onde a interblocagem pode ocorrer (não é implementada nenhuma estratégia de prevenção ou evitação).
- Periodicamente, o sistema recorre a um algoritmo que permite detectar um estado de interblocagem, identificando quais os processos e os recursos envolvidos.
- A execução do algoritmo e a estrutura de dados para suporte do mecanismo de detecção podem introduzir uma degradação significativa no desempenho do sistema.

# INTERBLOCAGEM: DETECÇÃO E REPARAÇÃO

Recuperação:

- Estratégia 1: terminação de processos
  - Terminar todos os processos envolvidos na interblocagem.
  - Terminar um processo de cada vez até que a interblocagem seja eliminada. A selecção do processo a terminar pode depender:
    - Prioridade do processo;
    - Tempo de processamento;
    - Recursos utilizados e recursos necessários para completar;
    - Número de processos que devem terminar;
    - Natureza do processo (interactivo ou batch).
- Estratégia 2: recorrer à preempção de recursos
  - Os processos devem estar preparados por forma a regressar (rollback) a um ponto de execução em que não existe interblocagem.
  - Por forma a evitar a privação, o número de vezes que um processo é sujeito à preempção de um recurso pode ser considerado como critério de selecção.