

# Sistemas Operativos

LEI - 2021/2022

**:: Deadlocks ::**

Escola Superior de Tecnologia de Setúbal - IPS

# Conteúdos

- Reconhecer deadlocks
- Métodos para prevenir e recuperar de deadlocks

# Deadlocks

Num deadlock os processos não terminam e os recursos ficam bloqueados, não permitindo que outros processos iniciem.

P1:

```
sem_wait(A)
sem_wait(B)
  .
  .
sem_post(B)
sem_post(A)
```

P2:

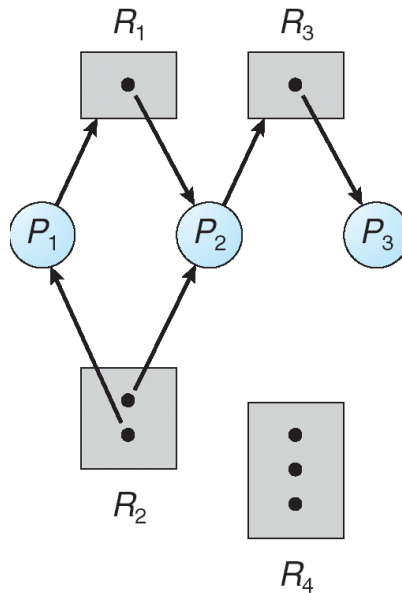
```
sem_wait(B)
sem_wait(A)
  .
  .
sem_post(A)
sem_post(B)
```

# Condições para deadlocks

1. **Exclusão mútua:** pelo menos um recurso deve ser bloqueado por um único processo
2. **Bloquear e esperar:** Um processo deve manter pelo menos um recurso e esperar por adquirir outros recursos
3. **Sem preempção:** Os recursos só podem ser libertos quando o processo assim o permitir
4. **Espera circular:** Deve existir um conjunto  $\{P_0, P_1, \dots, P_n\}$  de processos à espera, em que  $P_0$  espera por  $P_1$ ,  $P_1$  por  $P_2$ , etc.

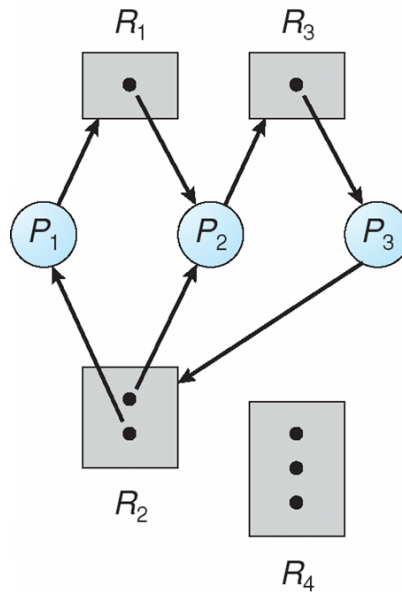
# Grafo de alocações

Ex:  $P = \{P_1, P_2, P_3\}$ ,  $R = \{R_1, R_2, R_3, R_4\}$ ,  
 $E = \{P_1 \rightarrow R_1, P_2 \rightarrow R_3, R_1 \rightarrow P_2, R_2 \rightarrow P_1, R_2 \rightarrow P_2, R_3 \rightarrow P_3\}$



 Sem ciclos, não há deadlock...

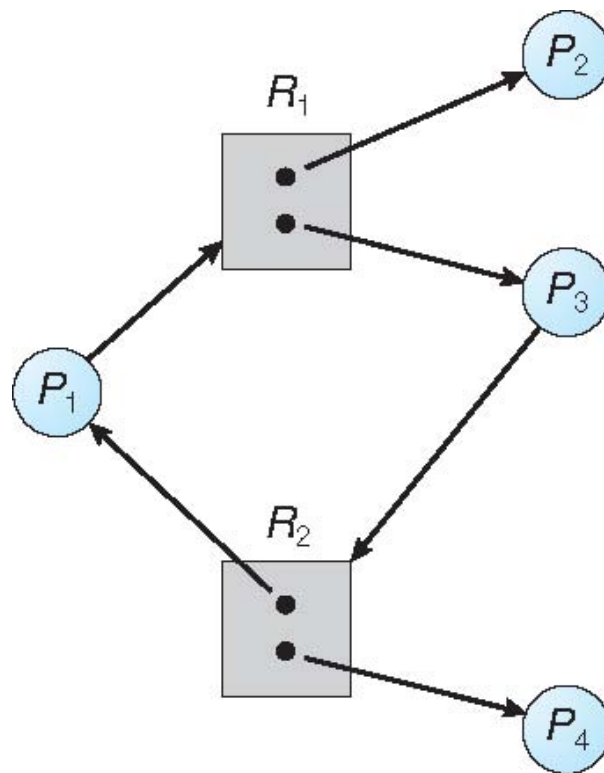
## Ex: P3 requer recurso R2



$P_1 \rightarrow R_1 \rightarrow P_2 \rightarrow R_3 \rightarrow P_3 \rightarrow R_2 \rightarrow P_1$   
 $P_2 \rightarrow R_3 \rightarrow P_3 \rightarrow R_2 \rightarrow P_2$

! Deadlock...


# Ciclos não implicam deadlocks..



💻 P4 pode libertar  $R_2$  e  $P_3$  pode então alocar  $R_2$ ..

# Métodos para prevenir deadlocks

- Assegurar que o sistema nunca entra num deadlock
- Permitir ao sistema recuperar de um deadlock
- Ignorar o problema e responsabilizar o programador

 A maioria dos sistemas operativos coloca a responsabilidade no lado do programador...



# Evitar deadlocks

Requer conhecimento *à priori* sobre o sistema..

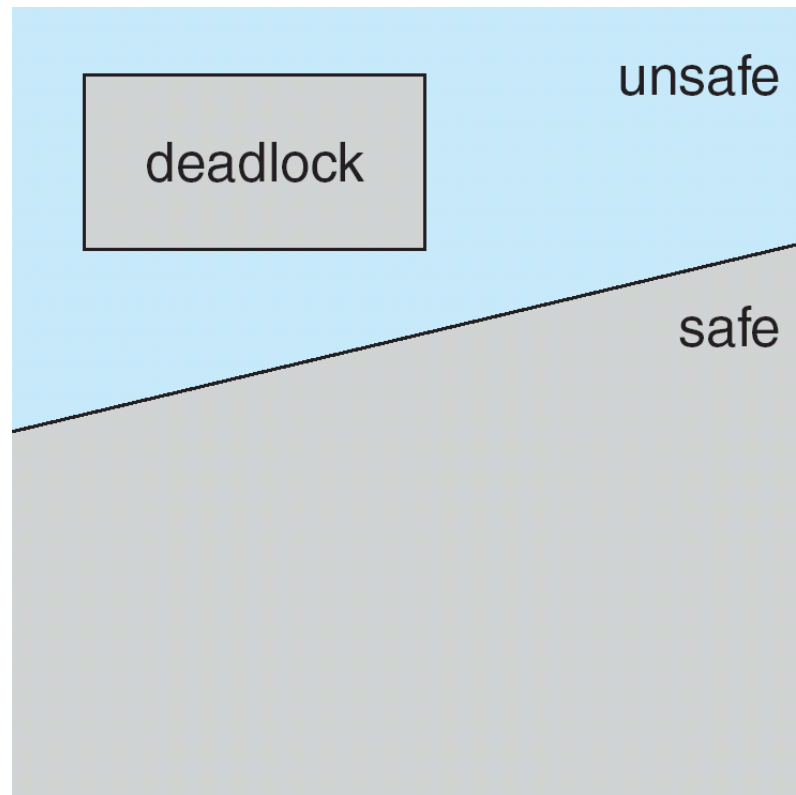
- Cada processo declara o número máximo de recursos que vai necessitar, de cada tipo
- O algoritmo examina os recursos de modo a evitar estados com ciclos

# Estado seguro

O sistema com processos  $\{P_1, \dots, P_n\}$  está num estado seguro se:

- Recursos que  $P_i$  necessita não estão disponíveis, pode esperar por processos  $P_j$
- Quando  $P_j$  termina,  $P_i$  consegue obter recursos
- Quando  $P_i$  termina,  $P_{i+1}$  consegue obter recursos de  $P_i$ .

Para  $j < i$ .

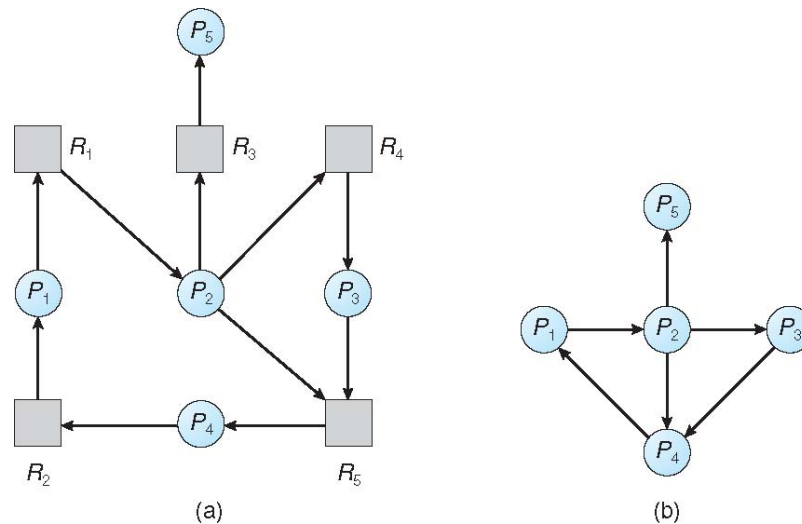



Estado seguro -> Sem deadlocks

Estado inseguro -> Possibilidade de deadlocks

# Detecção de deadlocks

Grafo de alocação de recursos e correspondente grafo de esperas



 Verificação de bloqueios consoante número de recursos disponíveis (ler secção 7.6).

# Recuperação de deadlocks

## Terminação de processos

- Abortar todos os processos bloqueados
- Abortar um processo de cada vez até eliminar o ciclo de deadlocks

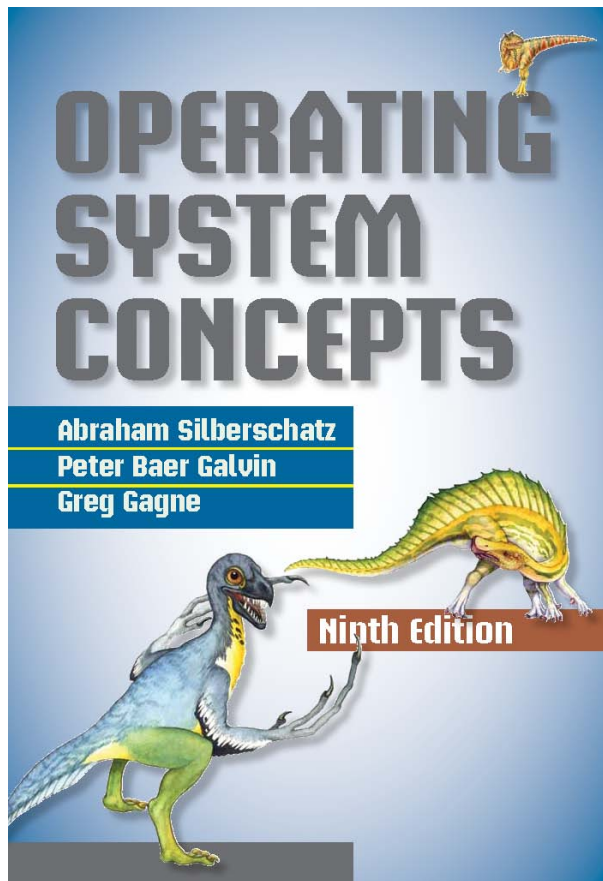
## Preempção de recursos

- Seleccionar processo: minimizar curto
- Rollback: retornar a um estado seguro
- Fome: o mesmo processo pode ser escolhido continuamente

**Quiz...**

# Sumário

- Deadlocks ocorrem quando dois ou mais processos bloqueiam à espera de recursos libertados por outros processos, também eles bloqueados
- Existem 3 formas de lidar com deadlocks
- Ignorar o problema e responsabilizar o programador é o mais comum



Ler capítulo 7...