Deadlocks

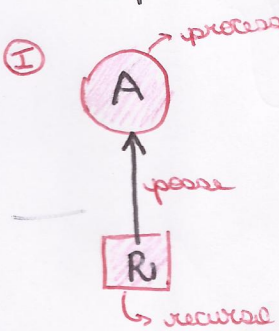
→ Recursos
- preemptível: memória ① → pode ser retirado sem afetar o processo
- não-preemptível: impressora ② → se retirar, afeta o processo

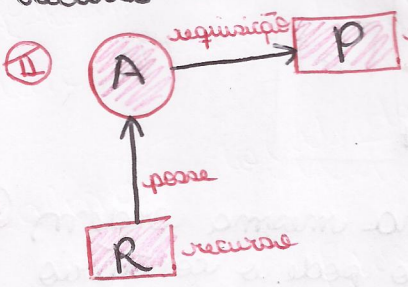
① Sejam dois processos A e B ambos precisam usar certos dados da memória para imprimir na impressora. Considere que A foi escolhido 1º e requisita a impressora, mas antes de efetivamente usá-la ocorre um time-out e B toma o lugar, pedindo a impressora. B requisita a memória e consegue, mas não a impressora. O que fazer?
↳ Toma-se a memória de B (→ copiando seus dados para o disco!) e permite que A continue seu processo, se terminar ele liberará a impressora será entregue a B.

② O processo A iniciou a impressão e ocorre um time-out. O B não poderá retirar dele.

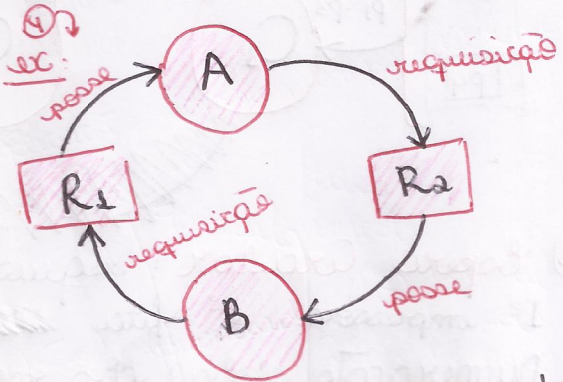
as 4 condições devem ocorrer para o surgimento de um deadlock

→ Condições
1. Exclusão mútua
2. Posse e Espera: processo A pega um recurso e fica esperando outro
3. Não-preempção: recursos não-preemptíveis
4. Espera Circular: cadeia circular de 2 ou mais processos, cada um esperando o recurso do outro.





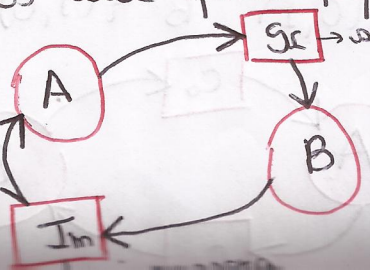
ex:



→ Como evitar?
↳ tirando uma das 4 condições

① Recuperação através da preempção → depois executar novamente
→ SPOOL: transferir dados p/ uma área temporária
↳ sacrificar uma tarefa p/ que as outras executem
* natureza do recurso (!)

① imprimir 10 páginas de um arquivo pdf



Solução: salvar uma memória e arquivo e a última página impressa no pdf de A, bloqueá-lo e deixar B executar. Acordar A.

② Recuperação através de voltar ao passado

↳ ficar guardando os estados da tarefa e de seus recursos periodicamente e se ocorrer um deadlock: voltar n processos necessários ANTES da requisição que causou o deadlock

Ataques:

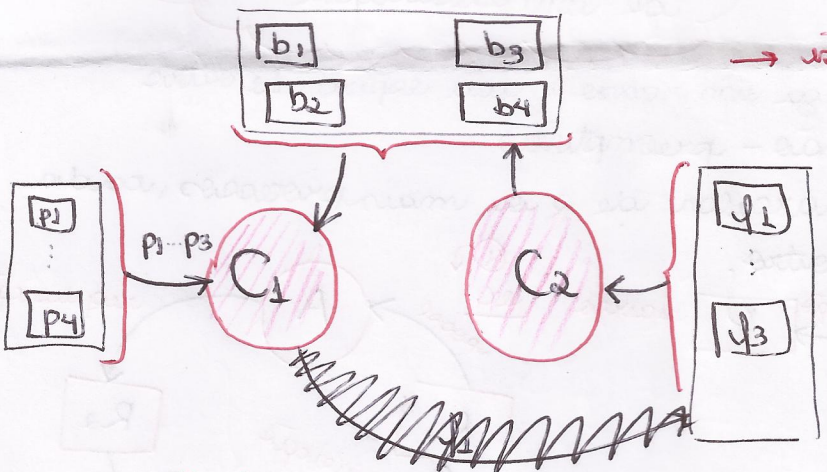
① Exclusividade Mútua: complicado. evitar entregar um recurso, quando ele não for absolutamente necessário, e tentar estar certo de que o mínimo possível de processos está precisando de recursos.

② Posse e da Espera:

⊕ alocar todos os recursos que precisa antes de iniciar a tarefa.

⊖ a tarefa, nem sempre, consegue saber todos os recursos que irá precisar. Acaba com o paralelismo

ex: C = cozinheiro p = panela b = boca do fogão f = frigideira



→ isso não daria certo!

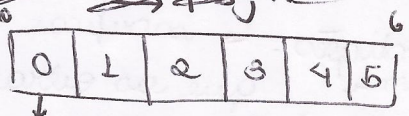
C2 vai ter que esperar C1 terminar tudo

③ Espera Circular: requisição na mesma ordem ①

① 1º: impressora 2º: fita → não pode o contrário

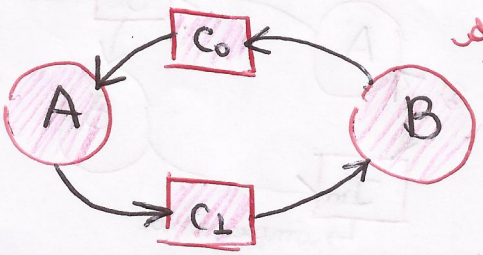
② Numeração global dos recursos. → recursos ⊕ (!) (i = R1, j = R2)

ex: Contas



transação (origem, destino)
lock1(m_or)
lock2(m_d)
//...//
unlock(m_d)
lock(m_or)

A: t(0,1) OR D B: t(1,0) OR D

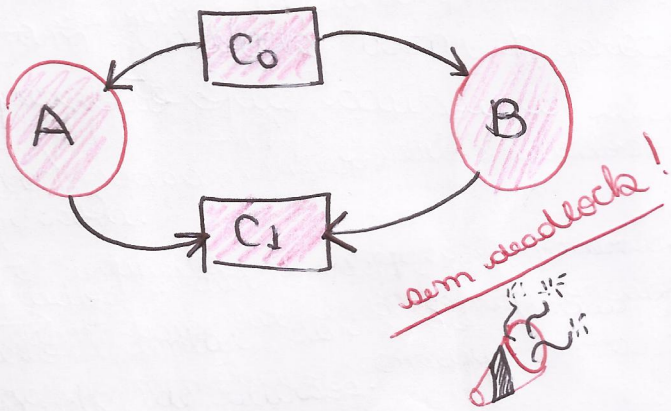


deadlock!

0	1	2	3	4	5

OR D — OR D

A: t(0,1) B: t(1,0)




```
if (id_or < id_d) (I)
    lock(m_or)
    lock(m_d)
      (...)
    unlock(m_d)
    unlock(m_or)
} else { (II)
    lock(m_d)
    lock(m_or)
      (...)
    unlock(m_or)
    unlock(m_d)
```