Sistemas Operacionais

Interação entre tarefas - impasses

Prof. Carlos Maziero

DInf UFPR, Curitiba PR

Fevereiro de 2019

Conteúdo

1. O conceito de impasse
2. Condições para impasse
3. Grafos de alocação de recursos
4. Estratégias de tratamento
   - Prevenção
   - Impedimento
   - Detecção e resolução

Impasses

- O coordenar tarefas: suspender algumas tarefas enquanto outras acessam recursos
- Cada recurso é associado a um semáforo ou algo equivalente
- As tarefas aguardam os semáforos para acessar os recursos
- Essas restrições podem levar a **impasses**

Impasse

Situação onde um grupo de tarefas fica bloqueado, aguardando umas pelas outras, reciprocamente.

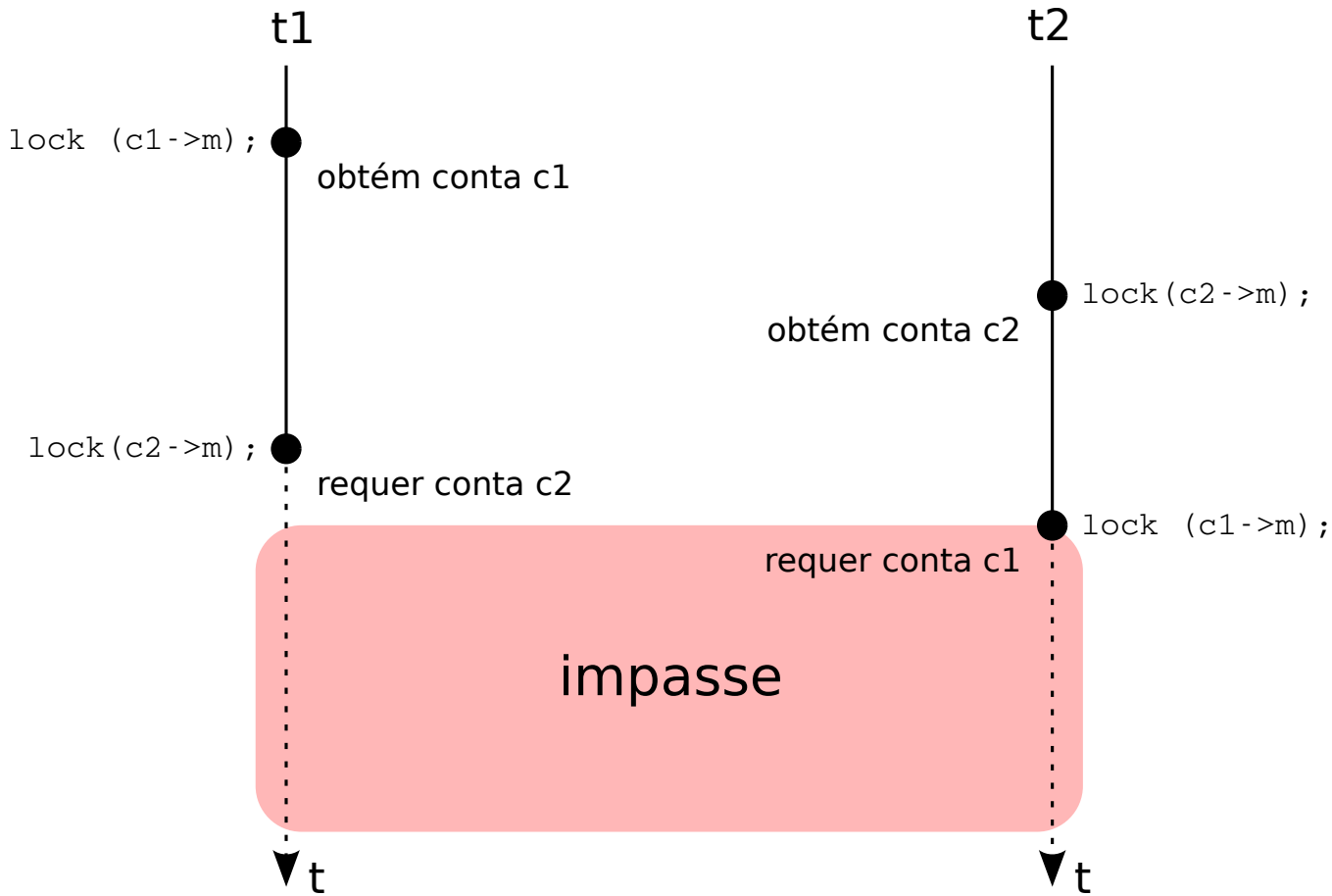
Exemplo de impasse

Operação de transferência bancária:

```
1  void transferir (conta_t* contaDeb, conta_t* contaCred, int valor)
2  {
3     sem_down (contaDeb->lock) ;   // obtém acesso a contaDeb
4     sem_down (contaCred->lock) ;  // obtém acesso a contCred
5
6     if (contaDeb->saldo >= valor)
7     {
8        contaDeb->saldo -= valor ; // debita valor de contaDeb
9        contaCred->saldo += valor ; // credita valor em contaCred
10    }
11    sem_up (contaDeb->lock) ;     // libera acesso a contaDeb
12    sem_up (contaCred->lock) ;    // libera acesso a contaCred
13 }
```

Exemplo de impasse

Dois clientes (tarefas t_1 e t_2) fazem transferências simultâneas entre suas contas ($c_1 \longrightarrow c_2$ e $c_2 \longrightarrow c_1$):



Um impasse real (São Paulo SP, 2017)

Condições para um impasse

Exclusão mútua : recursos acessados com exclusão mútua, gerida por semáforos ou similares

Posse e espera : a tarefa tem um recurso e quer acessar outro

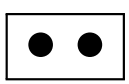
Não-preempção : a tarefa só libera seus recursos quando quiser, sem preempção

Espera circular : ciclo de esperas de recursos entre tarefas: a tarefa t_1 quer um recurso retido pela tarefa t_2, que quer um recurso retido por t_3, e assim por diante

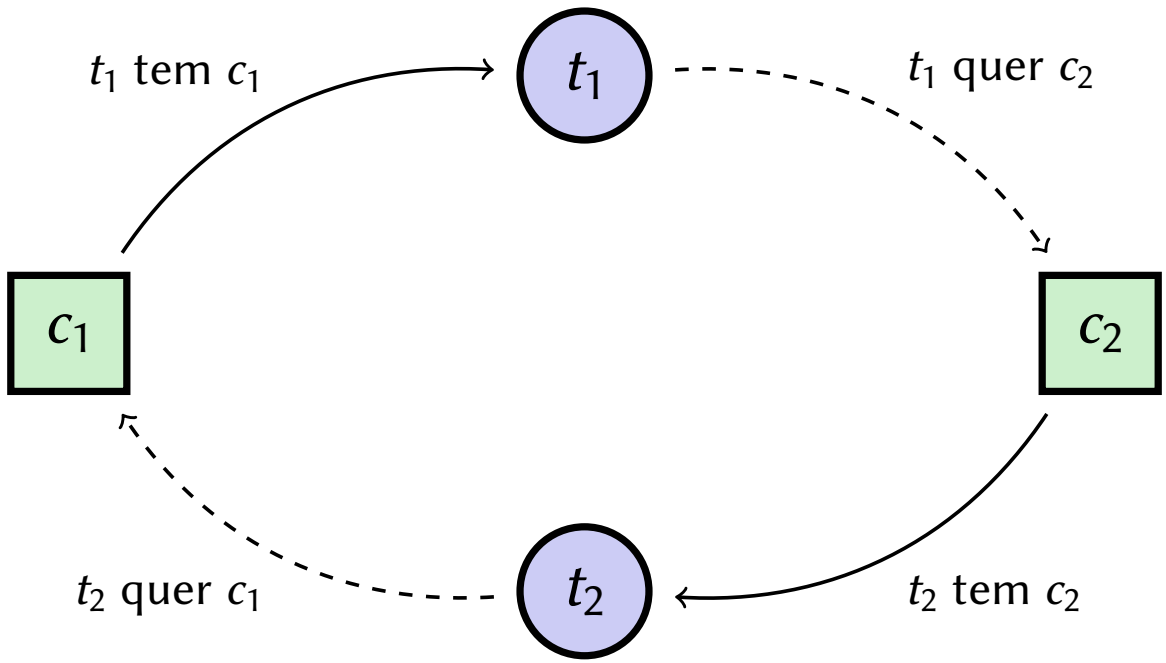
Estas quatro condições são **necessárias** (mas não suficientes) para a ocorrência de impasses.

Grafo de alocação de recursos

- permite visualizar a alocação de recursos
- permite detectar impasses

○	tarefa
□	recurso
□ → ○	posse de um recurso por uma tarefa (o recurso "pertence" à tarefa)
○ --> □	requisição de um recurso por uma tarefa (a tarefa "quer" o recurso)
	dois recursos do mesmo tipo

Impasse entre as contas bancárias



Percebe-se que as quatro condições estão presentes.

Impasse entre as contas bancárias

Percebe-se que as quatro condições estão presentes:

Exclusão mútua : as contas são protegidas por semáforos

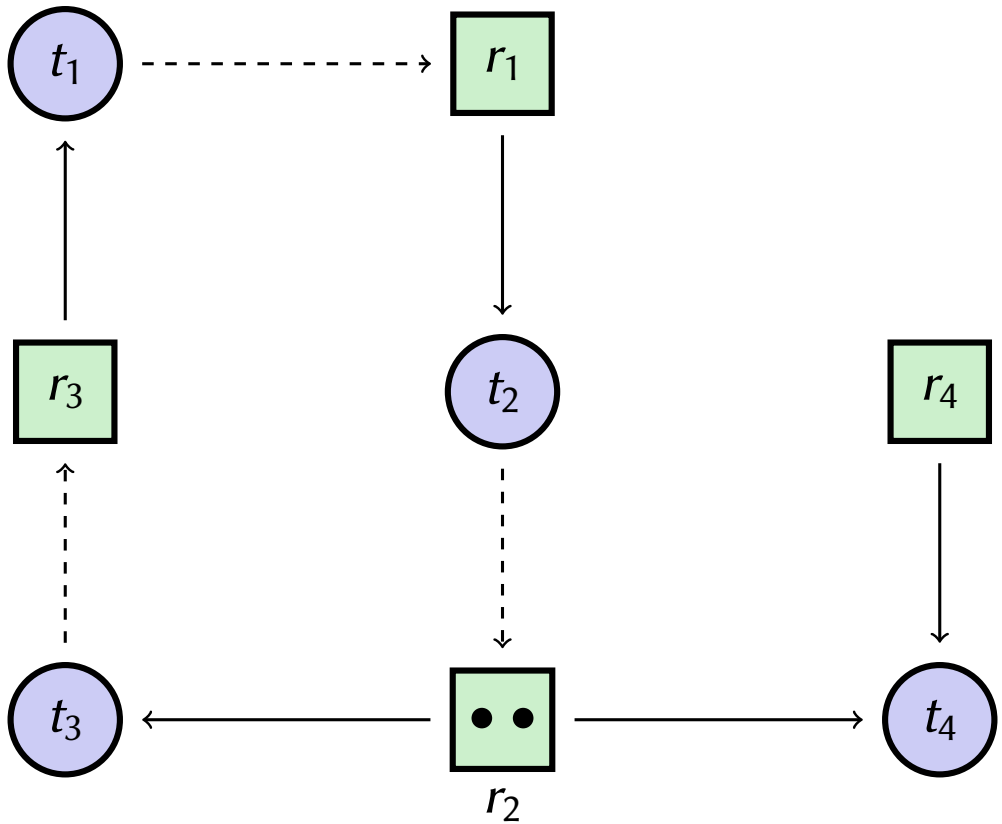
Posse e espera : requer conta c_i, requer conta c_j

Não-preempção : conta só é liberada com `sem_up(c)`

Espera circular : $c_1 \longrightarrow t_1 \dashrightarrow c_2 \longrightarrow t_2 \dashrightarrow c_1$

Quatro condições + um recurso de cada tipo $\longrightarrow$ `impasse!`

Múltiplas instâncias de recursos



Sem impasse, mesmo com $t_1 \dashrightarrow r_1 \longrightarrow t_2 \dashrightarrow r_2 \longrightarrow t_3 \dashrightarrow r_3 \longrightarrow t_1$

Impasses

Estratégias para tratar de impasses:

- **Ignorar** : ignorar o risco de impasses (problema do programador)
- **Prevenir** : regras de programação para evitar impasses
- **Impedir** : acompanhar a alocação de recursos e impedir impasses
- **Detectar** : detectar a ocorrência de impasses e desfazê-los

A mais usada em SOs: ignorar... 

Prevenção de impasses

Programar de forma a evitar as quatro condições:

Exclusão mútua : reduzir áreas de exclusão ao mínimo; usar técnicas alternativas, como o *spooling*

Posse e espera : Usar um recurso de cada vez, ou obter todos os recursos necessários antes de iniciar

Não-preempção : poder "arrancar" os recursos dos processos (difícil de implementar, pode gerar inconsistências)

Espera circular : os recursos são ordenados as tarefas os solicitam respeitando essa ordem

Basta quebrar **uma** condição!

Impedimento de impasses

Estratégia:

Acompanhar a alocação dos recursos às tarefas e negar acessos de recursos que possam levar a situações inseguras.

Grafo de estados do sistema:

Estado : uma distribuição de recursos entre as tarefas

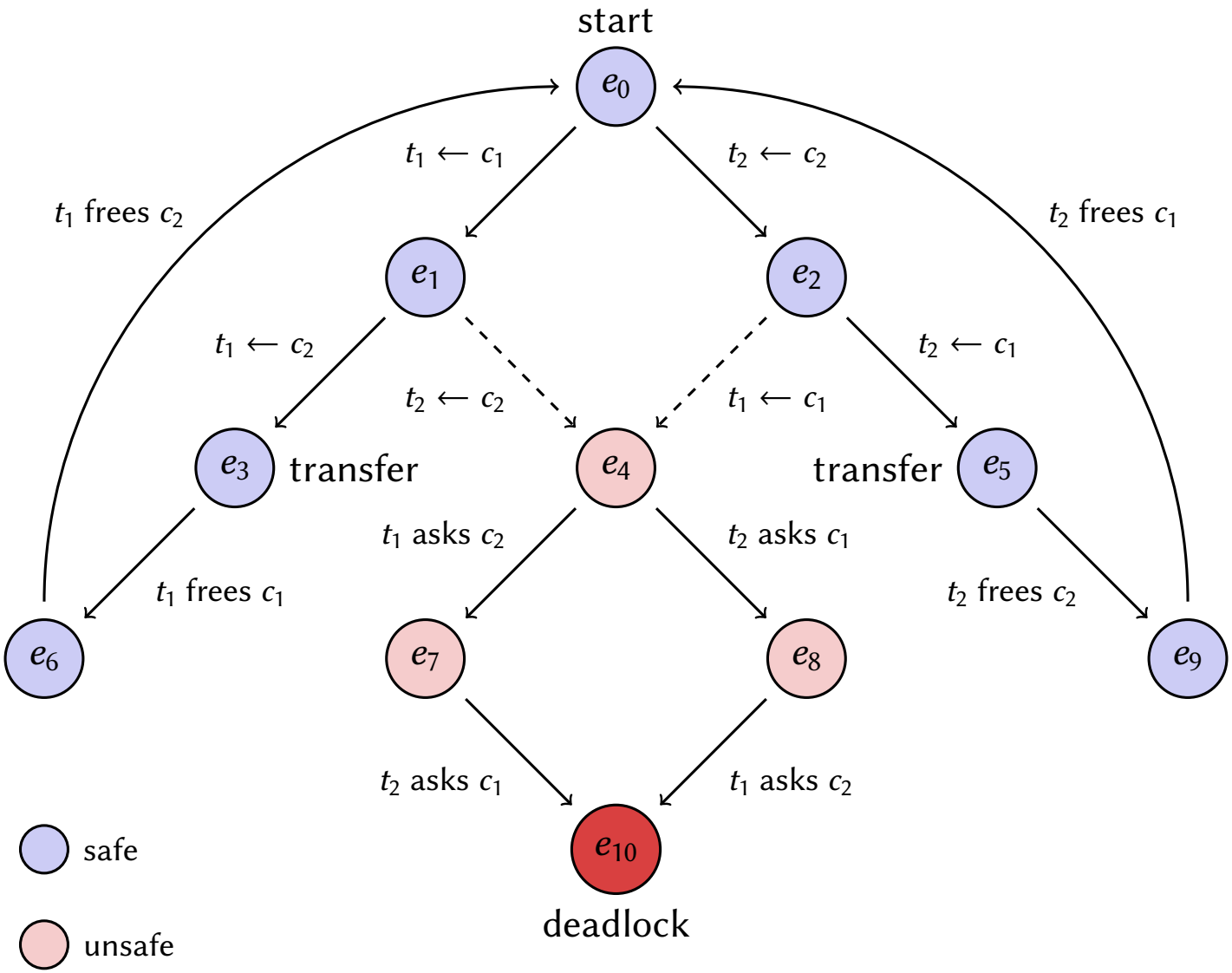
Aresta : uma alocação ou liberação de recursos

Os estados possíveis do sistema são divididos em:

Estados seguros : permitem evoluir aos demais estados

Estados inseguros : somente levam a impasses

Grafo de estados do sistema



Detecção e correção de impasses

Estratégia:

1. Observar o sistema
2. detectar a ocorrência de um impasse
3. Resolver o impasse

Como detectar impasses?

- Manter um grafo de alocação de recursos
- Detectar a formação de ciclos no grafo
- Pode exigir muito processamento

Detecção e correção de impasses

Como corrigir impasses?

- **Eliminar tarefas** de modo a romper os ciclos
  - que tarefas eliminar? A mais nova, a mais velha, a menos prioritária, ...
- **Retroceder tarefas**, retornando o sistema a um estado seguro
  - Técnica de *rollback* usada em bancos de dados
  - São necessários *checkpoints* periódicos
  - Algumas operações não podem ser desfeitas
    - interações com o usuário
    - envio de pacotes de rede