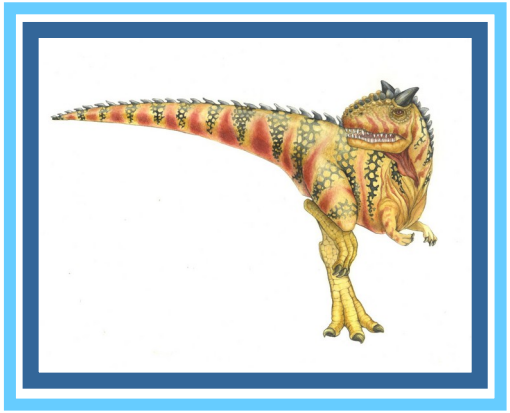
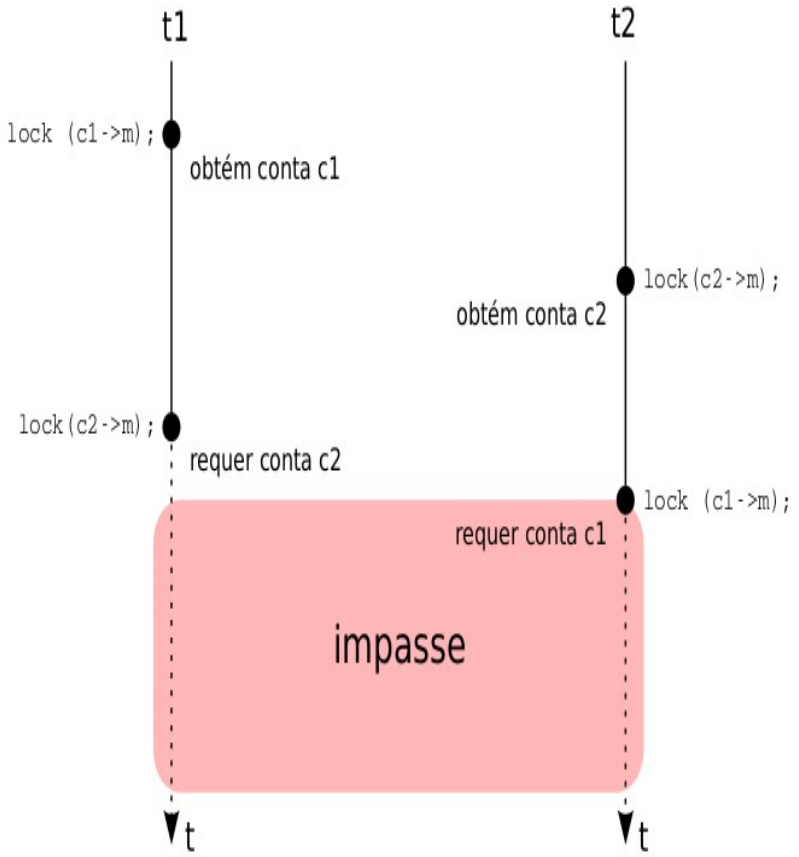
Deadlock



Deadlock

```
1  typedef struct conta_t
2  {
3     int saldo ;                    // saldo atual da conta
4     mutex m ;                      // mutex associado à conta
5     ...                            // outras informações da conta
6  } conta_t ;
7
8  void transferir (conta_t* contaDeb, conta_t* contaCred, int valor)
9  {
10    lock (contaDeb->m) ;           // obtém acesso a contaDeb
11    lock (contaCred->m) ;          // obtém acesso a contCred
12
13    if (contaDeb->saldo >= valor)
14    {
15       contaDeb->saldo -= valor ;  // debita valor de contaDeb
16       contaCred->saldo += valor ; // credita valor em contaCred
17    }
18    unlock (contaDeb->m) ;         // libera acesso a contaDeb
19    unlock (contaCred->m) ;        // libera acesso a contaCred
20 }
```



Deadlocks

Condições para impasses

Exclusão mútua: o acesso aos recursos deve ser feito de forma mutuamente exclusiva, controlada por semáforos ou mecanismos equivalentes. No exemplo da conta corrente, apenas uma tarefa por vez pode acessar cada conta.

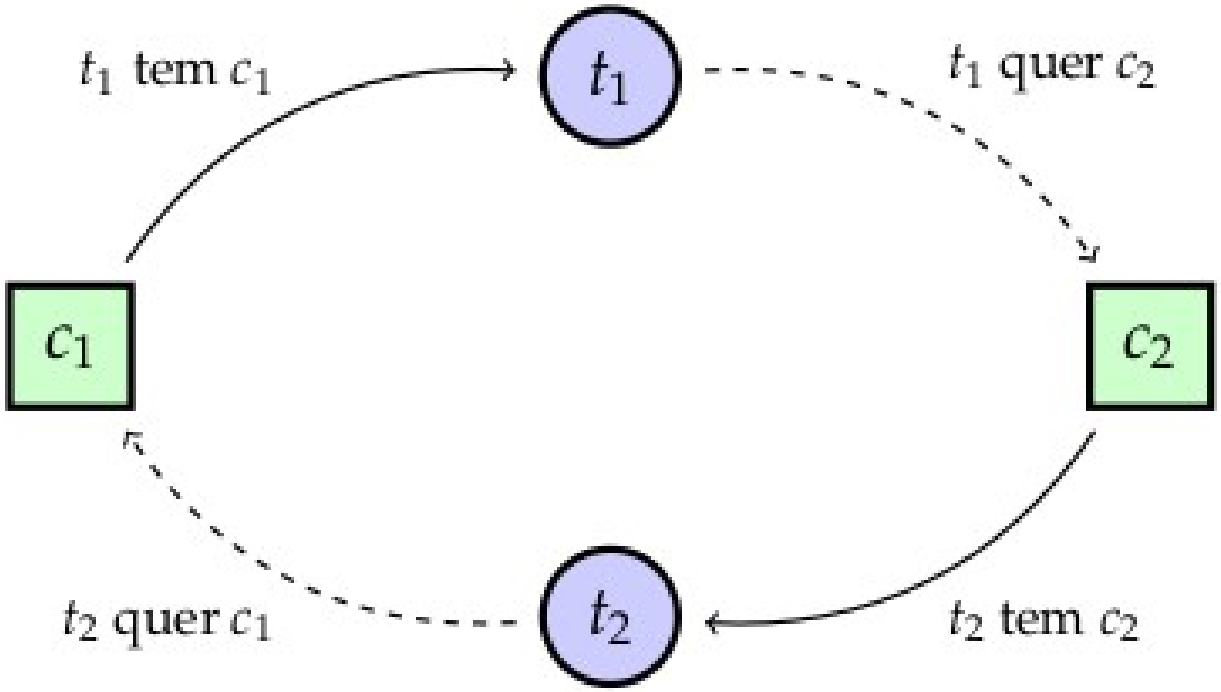
Posse e espera: uma tarefa pode solicitar o acesso a outros recursos sem ter de liberar os recursos que já detém. No exemplo da conta corrente, cada tarefa detém o semáforo de uma conta e solicita o semáforo da outra conta para poder prosseguir.

Não-preempção: uma tarefa somente libera os recursos que detém quando assim o decidir, e não os perde de forma imprevista (ou seja, o sistema operacional não retira à força os recursos alocados às tarefas). No exemplo da conta corrente, cada tarefa detém os *mutexes* obtidos até liberá-los explicitamente.

Espera circular: existe um ciclo de esperas pela liberação de recursos entre as tarefas envolvidas: a tarefa t_1 aguarda um recurso retido pela tarefa t_2 (formalmente, $t_1 \rightarrow t_2$), que aguarda um recurso retido pela tarefa t_3, e assim por diante, sendo que a tarefa t_n aguarda um recurso retido por t_1. Essa dependência circular pode ser expressa formalmente da seguinte forma: $t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow \cdots \rightarrow t_n \rightarrow t_1$. No exemplo da conta corrente, pode-se observar claramente que $t_1 \rightarrow t_2 \rightarrow t_1$.

Deadlocks

Grafos de alocação de recursos



Deadlocks

As principais técnicas usadas para tratar impasses em um sistema concorrente são:

- **prevenir impasses:** através, de regras rígidas para a programação dos sistemas,
- **impedir impasses:** por meio do acompanhamento contínuo da alocação dos recursos às tarefas,
- **detectar e resolver impasses**.