Sistemas Operacionais I

Aula passada

- Problemas Clássicos de Sincronização;
  - Produtores/Consumidores;
  - Leitores/Escritores;
  - O Jantar dos Selvagens;
  - O Jantar dos Filósofos.

Agenda

- Deadlock;
- Modelo de Sistema;
- Caracterização do Deadlock;
- Grafo de Alocação de Recursos.

Deadlock

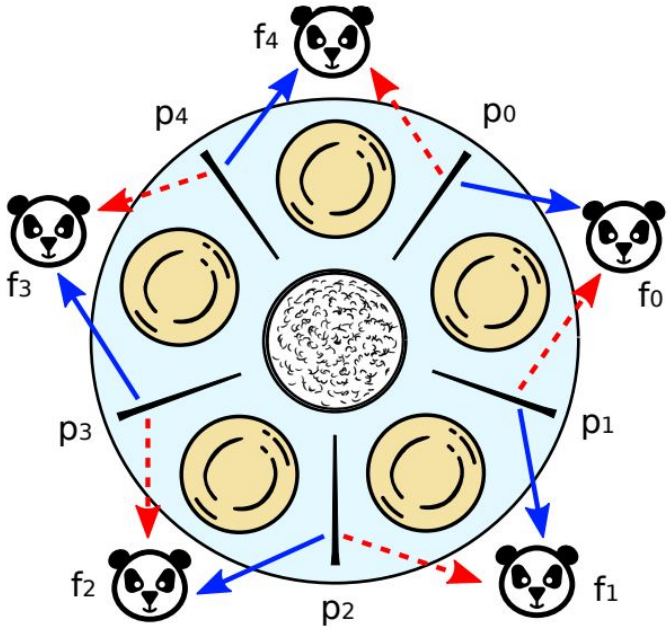
Em um ambiente de **multiprogramação**, vários processos podem **competir** por um número finito de recursos;

Um **processo** solicita **recursos**; se os recursos **não estão disponíveis** naquele momento, o processo entra em estado de **espera**;

Em alguns casos, um processo em espera não consegue mais mudar de estado novamente porque os recursos que ele solicitou estão reservados para outros processos em espera;

Essa situação é chamada **deadlock.**

Deadlock



Um **impasse** (**deadlock**) no jantar dos filósofos chineses.

Uma situação de **impasse** (**deadlock**) no trânsito

Modelo de Sistema

Modelo de Sistema

Um **sistema** é composto por um número **finito de recursos** a serem **distribuídos** entre vários **processos competidores**;

Os **recursos** podem ser divididos em vários **tipos** (ou **classes**), cada um composto por determinado número de **instâncias** idênticas;

Se um **sistema** tem **duas CPUs**, então o tipo de recurso CPU tem **duas instâncias**. Da mesma forma, o tipo de recurso impressora pode ter **cinco instâncias**.

Modelo de Sistema

Se um **processo** solicitar uma **instância** de um tipo de **recurso**, a **alocação** de qualquer **instância** do tipo deve **satisfazer à solicitação**. Se não satisfizer, as **instâncias** não são **idênticas**, e as **classes dos tipos de recursos não foram definidas apropriadamente**;

Por exemplo, um sistema pode ter **duas impressoras**. Essas duas impressoras podem ser definidas para estarem **na mesma classe de recursos** se ninguém se preocupar com qual impressora imprime que saída;

No entanto, se uma **impressora** estiver no **nono andar** e a outra estiver **no porão**, as pessoas no nono andar podem não considerar as duas impressoras como equivalentes, e classes de recurso separadas podem ter que ser definidas para cada impressora.

Modelo de Sistema

Um processo deve solicitar um recurso antes de usá-lo e deve liberar o recurso após usá-lo;

1. Solicitar o recurso;
2. Usar o recurso;
3. Liberar o recurso.

Solicitação: o processo solicita o recurso. Se a **solicitação não puder ser atendida** imediatamente (por exemplo, se o recurso estiver sendo usado por outro processo), o processo solicitante deve **esperar até que possa adquirir o recurso**;

Uso: o processo pode operar sobre o recurso (por exemplo, se o recurso for uma impressora, o processo pode imprimir na impressora);

Liberação: o processo libera o recurso.

Modelo de Sistema

Desenvolvedores de aplicações **multithreaded** devem ficar atentos à possibilidade de ocorrência de **deadlocks**.

As ferramentas de trancamento apresentadas (**semáforos** e **monitores**) foram projetadas para **evitar condições de corrida**;

No entanto, ao usar essas ferramentas, os **desenvolvedores** devem dar atenção especial a como os **locks** são adquiridos e **liberados**.

Caso contrário, podem ocorrer **deadlocks**, como ilustrado no problema dos filósofos;

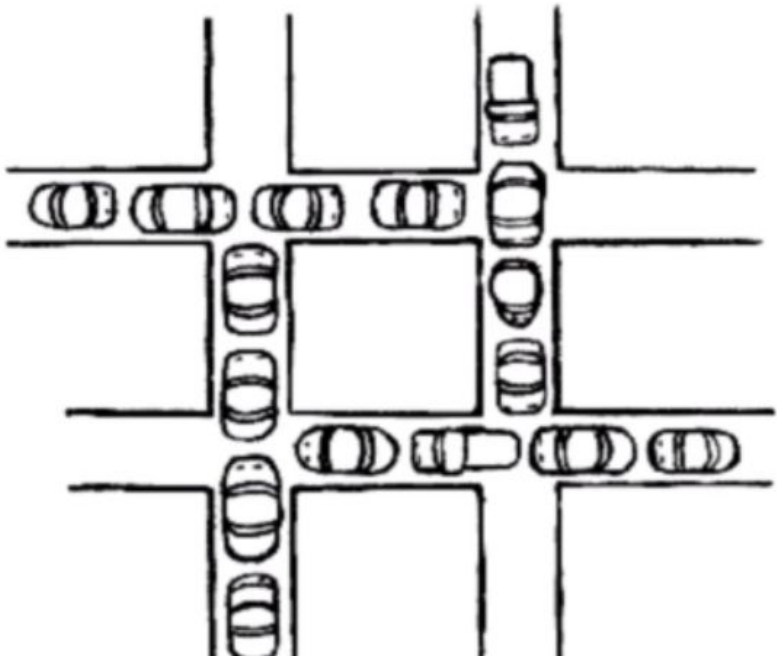
Caracterização do Deadlock

Caracterização do Deadlock

Uma situação de **deadlock** pode surgir se as **quatro condições** a seguir ocorrerem simultaneamente em um sistema:

1. **Exclusão mútua:** o acesso aos **recursos** deve ser feito de forma **mutuamente exclusiva**, controlada por **semáforos** ou **mecanismos equivalentes**.
2. **Posse e espera:** uma **tarefa** pode **solicitar** o acesso a **outros recursos** sem ter de **liberar** os recursos que já detém;
3. **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);
4. **Espera circular:** existe um ciclo de esperas pela liberação de recursos entre as tarefas envolvidas: **t1** → t2 → t3 → · · · → tn → **t1**.

Caracterização do Deadlock



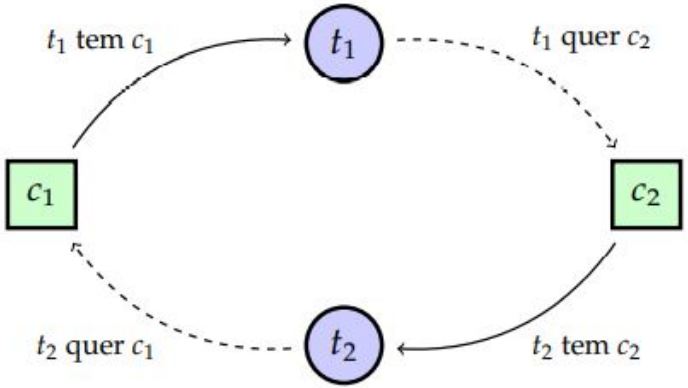
Espera circular no trânsito.

Grafo de Alocação de Recursos

Grafo de Alocação de Recursos

É possível representar **graficamente** a **alocação de recursos** entre as **tarefas** de um **sistema concorrente**;

A **representação gráfica** provê uma **visão mais clara da distribuição dos recursos** e permite detectar **visualmente** a presença de **esperas circulares** que podem caracterizar **deadlocks/impasses**;



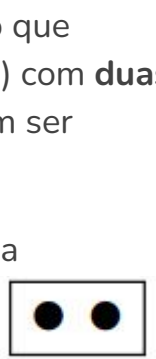
Grafo de alocação de recursos com impasse

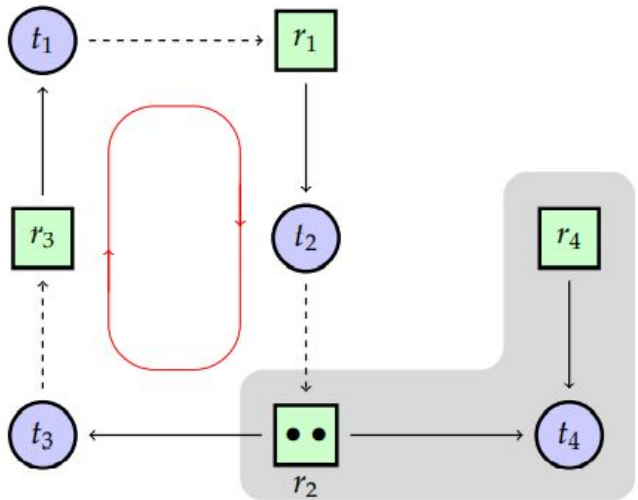
Grafo de Alocação de Recursos

Alguns **recursos** lógicos ou físicos de um **sistema computacional** podem ter **múltiplas instâncias**:

Por exemplo, um sistema pode ter **duas impressoras idênticas** instaladas, o que constituiria **um recurso** (impressora) com **duas instâncias equivalentes**, que podem ser **alocadas de forma independente**.

No **grafo de alocação de recursos**, a existência de **múltiplas instâncias** de um recurso é representada através de **"fichas" dentro dos retângulos**.





Grafo de Alocação de Recursos

É **importante** observar que a ocorrência de **ciclos em um grafo de alocação**, envolvendo **recursos com múltiplas instâncias**, pode **indicar** a presença de um **deadlock**, mas não garante sua existência.

Por exemplo, o ciclo
t1→ *r1* → *t2* → *r2* → *t3* → *r3* → *t1*
presente no diagrama não representa um **deadlock**, porque a qualquer momento a **tarefa** *t4* (que não está esperando recursos) pode liberar uma **instância do recurso** *r2*, solicitado por *t2*, permitindo atender a demanda de *t2* e desfazendo assim o **ciclo**.