COPPE UFRJ

GTA / UFRJ

Impasses

Pedro Cruz

EEL770 – Sistemas Operacionais

Lembretes

GTA / UFRJ

- Proposta de trabalho
  - Enviar para cruz@gta.ufrj.br
    - Prazo: hoje

Última aula

GTA / UFRJ

- Semáforos
  - *Up*
  - *Down*
- Mutexes
- Monitor
- Mensagens
  - Bloquear no envio/recebimento
  - Reconhecimento (*ack*)
  - Reenvio
    - Duplicatas

Hipótese de conclusão

GTA / UFRJ

- Se um processo tem todos os recursos que necessita, vai terminar em tempo finito

Recursos

GTA / UFRJ

- Preemptíveis
  - SO pode tomar de volta sem risco de problemas
    - Arquivo aberto em modo leitura

- Não-preemptíveis
  - Se forem tomados de volta, há risco de problemas
    - Arquivo aberto em modo escrita

Reserva de recursos

GTA / UFRJ

- Processo A
  - Reservar recurso X
  - Reservar recurso Y
  - .....
  - Liberar recurso X
  - Liberar recurso Y
- Processo B
  - Reservar recurso Y
  - Reservar recurso X
  - .....
  - Liberar recurso Y
  - Liberar recurso X

Uma execução possível

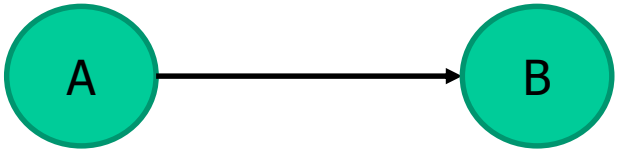
- Processo A precisa de recursos X e Y
- Processo B precisa dos recursos X e Y
- Processo A reserva recurso X
- Processo B reserva recurso Y
- Processo A aguarda recurso Y
- Processo B aguarda recurso X

E agora?

Representação

GTA / UFRJ

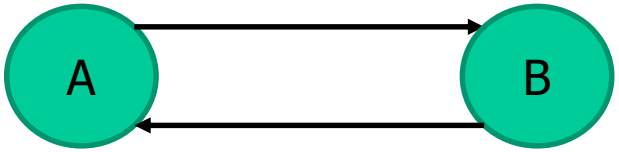
- Processo A aguarda que processo B libere algum recurso



Na representação

GTA / UFRJ

- Processo A aguarda que processo B libere algum recurso
- Processo B aguarda que processo A libere algum recurso



Condições para impasses

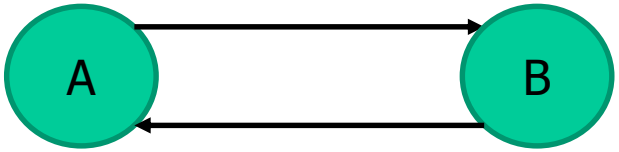
GTA / UFRJ

- Exclusão mútua
  - Processos precisam de acesso exclusivo a algum recurso
- Reservar e esperar
  - Processo pode reservar um recurso e aguardar outros
- Não preempção
  - Recurso reservado não poder ser tomado do processo
- Espera circular
  - Cadeia cíclica de reserva de recursos que o próximo precisa

Solução simples

GTA / UFRJ

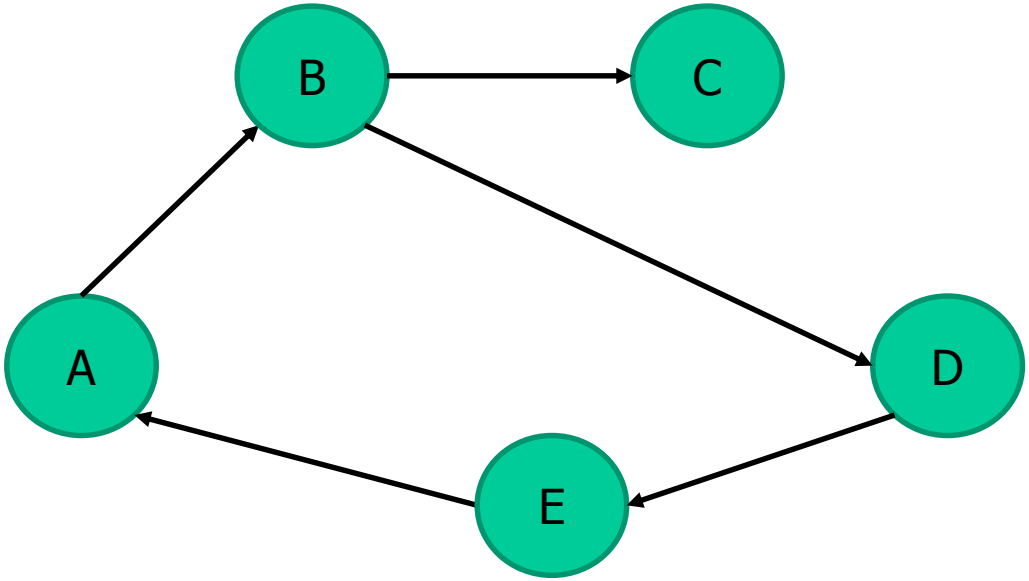
- Processo A envia mensagem pra B pedindo liberação
- Processo B consegue detectar *deadlock*



Casos piores

GTA / UFRJ

- Diversos processos estão em um impasse
  - E agora?



Resolução do problema de impasses

- Aceitar os impasses
- Detectar de impasses
- Prevenir impasses
- Evitar impasses

Aceitar impasses

GTA / UFRJ

- Qual o custo desse problema?
  - Custo de uma ocorrência
    - Reinicialização do sistema
    - Reinicialização de um conjunto de processos
  - Quantidade esperada de ocorrências
    - Uma vez ao dia?
    - Uma vez por uso?
- Qual o custo de solucionar esse problema?
  - Trabalho
  - Computação
  - Recursos

Algoritmo do avestruz

GTA / UFRJ

- Enfiar a cabeça na areia e deixar pra lá
  - Odiado por matemáticos
  - Amado por engenheiros
  - Utilizado quando vale a pena
    - Custo de impedir é maior do que o custo de reparar

Necessidade por recursos

GTA / UFRJ

- Processos podem precisar de muitos recursos
  - Mais de um tipo
  - Mais de uma unidade
- Recursos podem ser repetidos
  - Um processo precisa que apenas alguns estejam livres

Detecção de impasses - Grafo de esperas

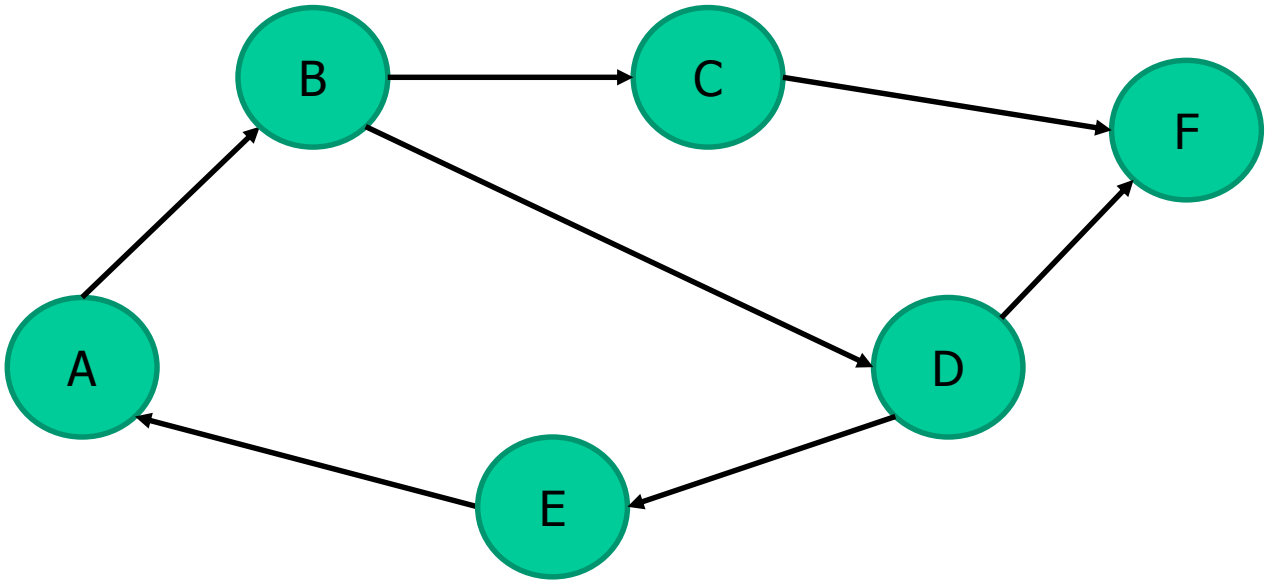
GTA / UFRJ

- Dois tipos de espera
  - Todos os vizinhos são necessários
    - Processo precisa de pelo menos um recurso
    - "E"
  - Qualquer vizinho basta
    - Processo precisa de todos os recursos
    - "OU"

Todos os vizinhos são necessários

GTA / UFRJ

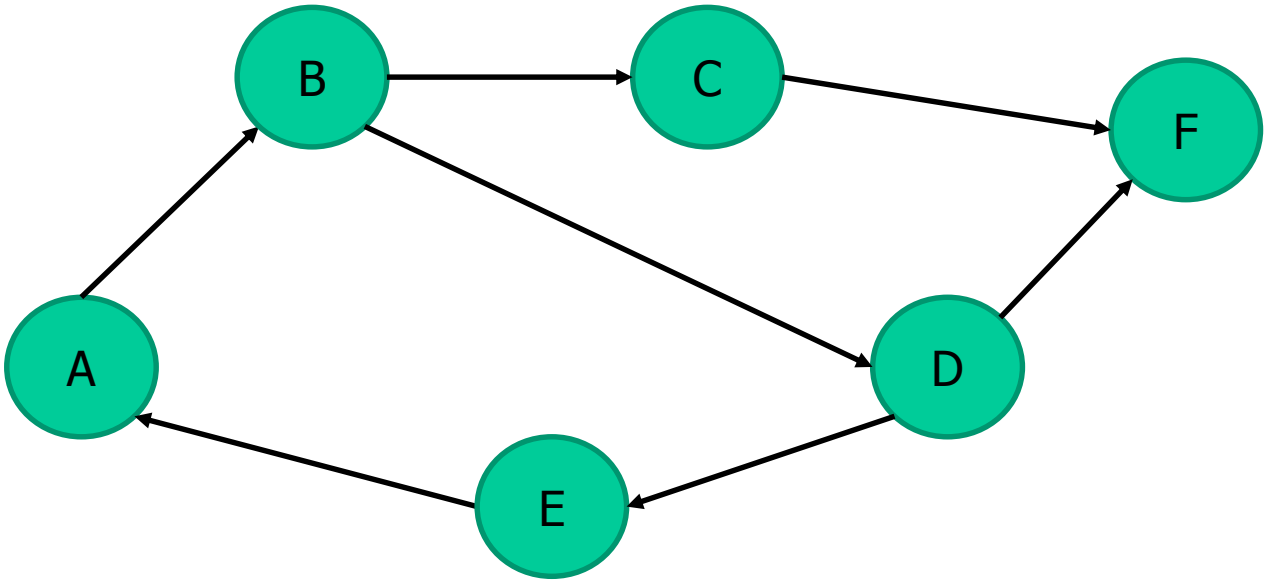
- Há impasse se há ciclo
  - Cada vértice no ciclo está esperando os outros do ciclo
    - Cada vértice no ciclo está esperando a si mesmo



Espera por todos os vizinhos

GTA / UFRJ

- Processo consegue terminar sua execução quando é sumidouro no grafo de esperas
  - Se existe caminho de um ciclo até um sumidouro, então, não existe impasse



Estratégias de recuperação

- Preempção
- Retrocesso
- Morte

Preempção

GTA / UFRJ

- Recursos "não-preemptíveis", *pero no mucho*
  - Precisa de cuidado
    - Algumas vezes, intervenção do usuário

Abordagem otimista

Retrocesso

GTA / UFRJ

- Processos criam pontos de salvaguarda
  - Salvaguarda representa um estado do processamento
    - Imagem da memória
    - Estado dos recursos
- Pontos de salvaguarda não são sobrescritos
- Detecção de impasse
  - Processo retorna ao estado representado pela salvaguarda

Abordagem pessimista

Morte

GTA / UFRJ

- Matar sempre processo cuja reexecução não seja danosa
  - Exemplo:
    - Reexecutar uma leitura de arquivo não causa danos
    - Reexecutar uma contagem pode causar danos

Evitando impasses

GTA / UFRJ

- Estado seguro
  - Existe ordem de escalonamento na qual todos os processos terminam sua execução
    - Mesmo que peçam todos os recursos a que têm direito
- Estado inseguro
  - Não existe escalonamento sem impasse se todos os processos pedirem os recursos a que têm direito

Exemplo de estados seguros/inseguros

GTA / UFRJ

Máximo de recursos:
Total: 10
Por processo: 5

Recursos com A	Recursos com B	Recursos com C	Recursos disponíveis	Estado
0	0	0	10	Seguro
3	1	1	5	Seguro
4	1	1	4	Seguro
4	2	1	3	Inseguro

Evitando impasses

GTA / UFRJ

- Evitar estados inseguros
  - Algoritmo do banqueiro

Evitando impasses – algoritmo do banqueiro

- Problema:
  - Banqueiro precisa garantir que haja dinheiro no banco para cada cliente que tem direito de sacar dinheiro
  - Banqueiro quer emprestar dinheiro aos clientes

- Solução:
  - Banqueiro empresta, no máximo, quantidade de dinheiro de forma que consiga devolver dinheiro para o cliente que mais pode sacar

Algoritmo do banqueiro - Dijkstra

- Ao receber um pedido de recurso
  - O atendimento geraria um estado inseguro?
    - Sim:
      - Impedir o atendimento e bloquear o processo
    - Não:
      - Atender ao pedido
- Ao receber um recurso de volta
  - Algum processo bloqueado pode ter pedido atendido?
    - Sim:
      - Atender ao pedido
    - Não:
      - Deixar pra lá

Problemas com o algoritmo do banqueiro

- A fila de processos bloqueados deve ser gerenciada de forma sábia
  - Evitar que processo fique esperando "infinitamente"

- Na prática, processos não sabem quanto de recursos precisarão

Prevenção de impasses

GTA / UFRJ

- Atacar as condições de impasses
  - Exclusão mútua
    - Evitar operações com exclusão mútua
  - Reserva e espera
    - Processos recebem tudo ou nada do que precisam
  - Não-preempção
    - Virtualizar recursos
  - Espera circular
    - Recursos possuem ordem para serem pedidos
      - Gera uma árvore, logo, não há ciclos

Um adendo

GTA / UFRJ

- Travamento em duas fases
  - Processo tenta obter seus recursos, um de cada vez
  - Se um dos recursos não estiver disponível, devolve todos

- Devolução obrigatória
  - Processo obtém recursos
    - Para obter novos, primeiros devem ser devolvidos

Processos "educados"

GTA / UFRJ

- Processos "educados"
  - Devolvem recursos quando não recebem todos que precisam
  - Aguardam um pouco e tentam novamente

- Processo A
  - Reservar recurso X
  - Reservar recurso Y
    - Negado
    - Devolver X
    - Tentar novamente
  - Utilizar recursos

- Processo B
  - Reservar recurso Y
  - Reservar recurso X
    - Negado
    - Devolver Y
    - Tentar novamente
  - Utilizar recursos

Livelock

GTA / UFRJ

- Processos competem por recursos
  - Devolvem recursos quando não podem reservar todos
  - Nenhum processo consegue reservar todos

Inanição

GTA / UFRJ

- Política dá sempre recurso pra quem precisa de menos
  - Aumenta a eficiência do sistema
    - O que acontece com processo que precisa de muitos recursos?
    - Há impasse?

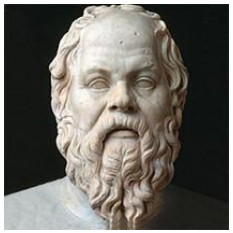
Jantar dos filósofos - Dijkstra

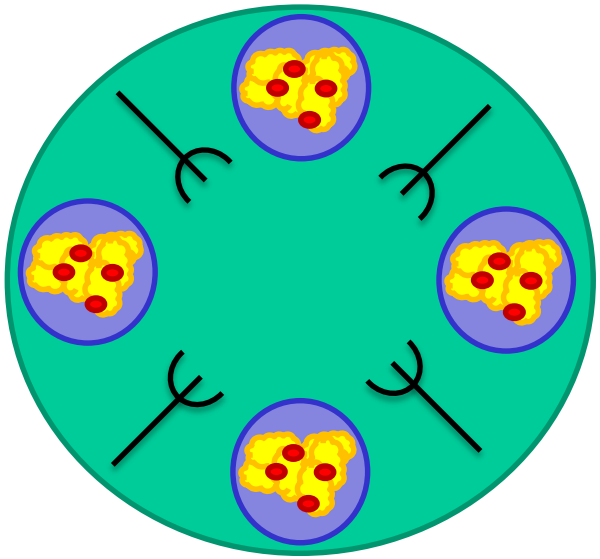
GTA / UFRJ

- Filósofos estão jantando em uma mesa circular
  - Pensam
  - Ficam com fome
  - Comem
  - Voltam a pensar
- Filósofos precisam de dois talheres
- Filósofos não se falam

Jantar dos filósofos

GTA / UFRJ





Jantar dos filósofos - problemas

GTA / UFRJ

- Como garantir exclusão mútua?
  - Pode haver impasse?
  - Pode haver livelock?
- Como garantir que nenhum filósofo morra de fome?
  - Nenhum filósofo pode segurar garfo pra sempre
  - Nenhum filósofo pode ter seus dois vizinhos alternando

- Como fazer com que os filósofos esperem o mínimo possível com fome?

Soluções comuns

GTA / UFRJ

- Numerar os garfos
  - Todo mundo pega o garfo de número menor primeiro
    - Pouco paralelismo
- Vez
  - Filósofo come e passa a vez para o da esquerda
    - Precisa de uma inicialização "correta"
- Ditador
  - Um líder define quem pode comer
    - Pode haver diferentes níveis de paralelismo
- Gentileza
  - Filósofos tentam pegar garfos
    - Soltam se não puderem usar os dois

COPPE UFRJ

GTA / UFRJ

Impasses

Pedro Cruz

EEL770 – Sistemas Operacionais