Sistemas Operacionais

Prevenção e outras questões de deadlocks

Profª Drª Thaína Aparecida Azevedo Tosta

tosta.thaina@unifesp.br

UNIFESP
Universidade Federal de São Paulo

São José dos Campos
2024/01

Aula passada

- Detecção de impasses
- Recuperação de um impasse
- Evitando impasses

Sumário

- Prevenção de impasses
- Outras questões de impasses

Objetivo: questão de fixação.

Prevenção de impasses

- Como os sistemas evitam impasses se não é possível ter informações futuras?
- Assegurar que uma das condições **não** seja satisfeita:

1. Condição de exclusão mútua. Cada recurso está atualmente associado a exatamente um processo ou está disponível;
2. Condição de posse e espera. Processos atualmente de posse de recursos que foram concedidos antes podem solicitar novos recursos;
3. Condição de não preempção. Recursos concedidos antes não podem ser tomados à força de um processo. Eles precisam ser explicitamente liberados pelo processo que os tem;
4. Condição de espera circular. Deve haver uma lista circular de dois ou mais processos, cada um deles esperando por um processo de posse do membro seguinte da cadeia.

Prevenção de impasses

Condição de exclusão mútua

- Se nunca acontecer de um recurso ser alocado exclusivamente para um único processo, jamais teremos impasses;
- Para dados: apenas leitura;
- Para recursos: permitir que dois processos escrevam na impressora ao mesmo tempo levará ao caos → técnica de spooling e daemon.

Prevenção de impasses

Condição de exclusão mútua

- Daemons são programados para imprimir somente após o arquivo de saída completo estar disponível;
- Se dois processos ainda não tiverem completado suas saídas, mesmo preenchendo metade do espaço disponível de spool com suas saídas? Impasse no disco!
- **Evitar alocar um recurso a não ser que seja absolutamente necessário (quanto menos, melhor).**

Prevenção de impasses

Atacando a condição de posse e espera

- Se pudermos evitar que processos que já possuem recursos esperem por mais recursos, poderemos eliminar os impasses;
- Uma maneira é exigir que todos os processos solicitem todos os seus recursos antes de iniciar a execução;
- Funciona?

Prevenção de impasses

Atacando a condição de posse e espera

- Problema 1: muitos processos não sabem quantos recursos eles precisarão até começarem a executar;
- Problema 2: os recursos não serão usados de maneira otimizada;
- Para quem serve? Sistemas em lote;
- E agora? Exigir que um processo que solicita um recurso primeiro libere temporariamente todos os recursos atualmente em suas mãos.

Prevenção de impasses

Atacando a condição de não preempção

- Se a impressora foi alocada a um processo e ele está no meio da impressão, podemos tomar à força a impressora porque uma plotter de que esse processo também necessita não está disponível;
- Possibilidade: promover o spooling da impressora e permitir que apenas o daemon acesse a impressora real;
- Nem todos os recursos podem ser virtualizados, como bancos de dados ou tabelas do SO.

Prevenção de impasses

Atacando a condição da espera circular

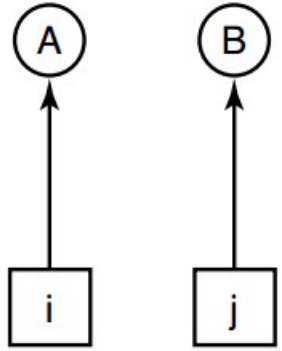
- Uma forma de eliminar a espera circular é ter uma regra dizendo que um processo tem o direito a apenas um único recurso de cada vez;
- Se ele precisar de um segundo recurso, precisa liberar o primeiro;
- Problema?

Prevenção de impasses

Atacando a condição da espera circular

- Outra maneira é uma numeração global de todos os recursos, onde as solicitações precisam ser feitas em ordem numérica;
- Impasse se A solicita j e B solicita i;
  - Se i > j, então A não tem permissão para solicitar;
  - Se i < j, então B não tem permissão para solicitar.
- Problemas?

1. Máquina de fotolito
2. Impressora
3. Plotter
4. Unidade de fita
5. Unidade de Blu-ray



Prevenção de impasses

Condição	Abordagem contra impasses
Exclusão mútua	Usar spool em tudo
Posse e espera	Requisitar todos os recursos necessários no início
Não preempção	Retomar os recursos alocados
Espera circular	Ordenar numericamente os recursos

Outras questões

Travamento em duas fases (*two-phase locking*)

- Em bancos de dados, é frequente solicitar travas em vários registros e então atualizá-los;
- Fase 1: o processo tenta travar todos os registros de que precisa, um de cada vez;
- Fase 2: o processo atualiza e libera as travas;
- Se algum registro travado é solicitado, o processo pode liberar todas as travas e começar a fase 1 desde o início, ou ignorar a solicitação;
- Problemas?

Outras questões

Travamento em duas fases (*two-phase locking*)

- Em sistemas de tempo real e sistemas de controle de processos, por exemplo, não é aceitável:
  - Terminar um processo e começar tudo de novo;
  - Reiniciar se o processo já leu ou escreveu mensagens para a rede, arquivos atualizados ou qualquer coisa que não possa ser repetida seguramente.
- Quando funciona? Quando o programa pode ser parado em qualquer ponto durante a primeira fase e reiniciado.

Outras questões

Impasses de comunicação

Um processo quer algo que outro processo tem (hardware ou software) e deve esperar até que o primeiro abra mão dele → impasses de recursos → **sincronização de competição** (processos independentes completariam seus serviços se a sua execução não sofresse a competição de outros processos).

```
semaphore resource_1;
semaphore resource_2;

void process_A(void) {
        down(&resource_1);
        down(&resource_2);
        use_both_resources( );
        up(&resource_2);
        up(&resource_1);
}

void process_B(void) {
        down(&resource_2);
        down(&resource_1);
        use_both_resources( );
        up(&resource_1);
        up(&resource_2);
}
```

Outras questões

Impasses de comunicação

- Impasses de recursos são os mais comuns, mas não os únicos;
- Em sistemas de comunicação, dois ou mais processos comunicam-se enviando mensagens:
  - O processo A envia uma mensagem de solicitação a B, e bloqueia até B enviar uma resposta;
  - Se a mensagem de solicitação se perder? Impasse de comunicação (anomalia de sincronização de cooperação → os processos não podem completar se executados independentemente);
  - Solução?

Outras questões

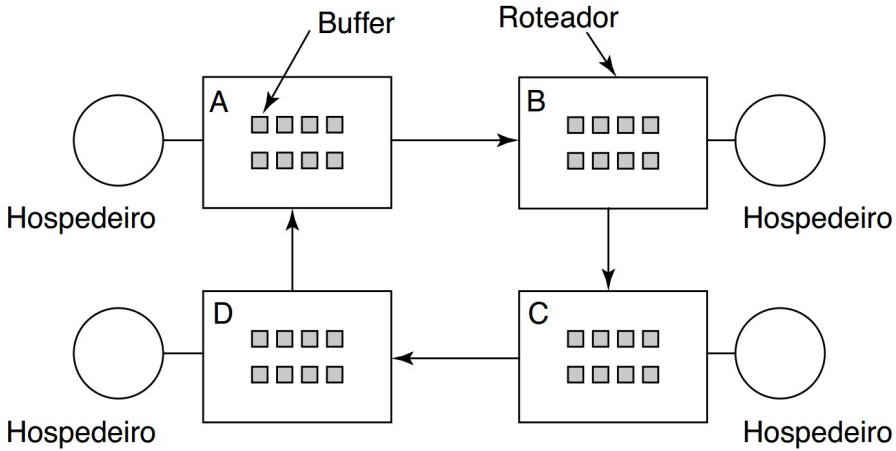
Impasses de comunicação

- A técnica que pode ser empregada para acabar com os impasses de comunicação são os controles de limite de tempo (*timeouts*);
- Sempre que uma mensagem é enviada para a qual uma resposta é esperada, um temporizador é inicializado;
- Consequências? Como lidar com elas? Regras de comunicação (protocolos).

Outras questões

Impasses de comunicação

- Nem todos os impasses que ocorrem em sistemas de comunicação ou redes são impasses de comunicação. Impasses de recursos também acontecem;
- Quando um pacote chega a um roteador vindo de um dos seus hospedeiros, ele é colocado em um buffer para transmissão subsequente para outro roteador e então outro até que chegue ao destino.



Outras questões

Livelock

- Em algumas situações, um processo pode liberar os bloqueios que ele já adquiriu sempre que nota que não pode obter o bloqueio seguinte de que precisa;
- Então ele espera um milissegundo, digamos, e tenta de novo;
- E se o outro processo faz a mesma coisa no mesmo momento?

Outras questões

Livelock

Nenhum processo é bloqueado e poderíamos até dizer que as coisas estão acontecendo, então isso não é um impasse (deadlock *vs* livelock).

```
void process_A(void) {
        acquire_lock(&resource_1);
        while (try_lock(&resource_2) == FAIL) {
                release_lock(&resource_1);
                wait_fixed_time();
                acquire_lock(&resource_1);
        }
        use_both_resources( );
        release_lock(&resource_2);
        release_lock(&resource_1);
}
```

```
void process_B(void) {
        acquire_lock(&resource_2);
        while (try_lock(&resource_1) == FAIL) {
                release_lock(&resource_2);
                wait_fixed_time();
                acquire_lock(&resource_2);
        }
        use_both_resources( );
        release_lock(&resource_1);
        release_lock(&resource_2);
}
```

Outras questões

Livelock

- Livelocks e impasses podem ocorrer de maneiras surpreendentes:
  - O número total de processos permitidos é determinado pelo número de entradas na tabela de processos;
  - O número máximo de arquivos abertos é similarmente restrito pelo tamanho da tabela de i-nós;
  - Quase todas as tabelas no sistema operacional representam um recurso finito.
- O que fazer?

Outras questões

Livelock

- A maioria dos sistemas operacionais, incluindo UNIX e Windows, na essência apenas ignora o problema;
- Livelock ocasional (ou mesmo um impasse) é melhor que uma regra restringindo todos os usuários;
- Uma solução envolve aplicação de restrições inconvenientes considerando o que é mais importante, e para quem.

Outras questões

Inanição

- Um problema próximo ao impasse e ao livelock é a inanição (*starvation*);
- Alguma política é necessária sobre quem recebe qual recurso e quando;
- Exemplo:
  - Alocação de impressora para processo com o menor arquivo;
  - O que acontece em um sistema ocupado quando um processo tem um arquivo enorme para imprimir? Postergado indefinidamente, mas não bloqueado.

Outras questões

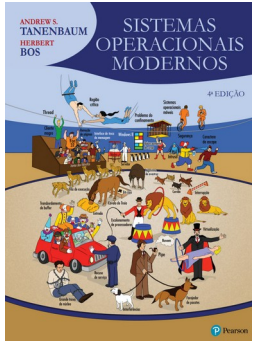
Inanição

- A inanição pode ser evitada com uma política de alocação de recursos primeiro a chegar, primeiro a ser servido;
- Algumas pessoas não fazem distinção entre a inanição e o impasse, porque em ambos os casos não há um progresso;
- Outras creem tratar-se de conceitos fundamentalmente diferentes.

Objetivo: questão de fixação

Um estudante de ciência da computação designado para trabalhar com impasses pensa na seguinte maneira brilhante de eliminar os impasses. Quando um processo solicita um recurso, ele especifica um limite de tempo. Se o processo bloqueia porque o recurso não está disponível, um temporizador é inicializado. Se o limite de tempo for excedido, o processo é liberado e pode executar novamente. Se você fosse o professor, qual nota daria a essa proposta e por quê?

Referências



TANENBAUM, Andrew S.; BOS, Herbert. Sistemas operacionais modernos. 4. edição. São Paulo: Pearson, 2016. xviii, 758 p. ISBN 9788543005676.

O modelo desta apresentação foi criado pelo Slidesgo.
Agradeço ao Prof. Bruno Kimura da Universidade Federal de São Paulo pelo material disponibilizado.