Capítuo 6 – Impasses

- Os sistemas operacionais devem ser capazes de garantir o acesso exclusivo a certos recursos.
- Impasses podem ocorrer tanto em recursos de hardware quanto em recursos de software.

6.1 – Recursos

- Impasses ocorrem quando vários processos recebem o direito exclusivo obre um recurso.
- Um recurso pe algo que pode ser adiquirido, utilizado e liberado com o passar do tempo..

6.1.1 – Recursos preemptíveis e não preemptíveis.

- Um recruso preemptivel pode ser retirado do processo proprietário sem nenhum prejuízo. A memória é um exemplo.
- Um recurso não preemptível não pode ser retirado do processo proprietário.
- Em geral impasses envolvem recursos não preemptíveis.

6.1.2 – Aquisição de recursos

- A ordem de solicitação dos recursos importa.

6.2 – Introdução aos impasses

- Um impasse é um conjunto de processos onde todos os processos desse conjunto estão esperando por um recursos que só está disponível através de outros processo desse mesmo conjunto.

6.2.1 – Condições para ocorrência de impassede recursos.

- Há quatro condições:
    - Condição de exclusão mútua. Em um determinado instante um recurso pode está apenas ou associado à um processo ou disponível.
    - Condição de posse e espera.Recursos que em um determinado instante retem recursos podem requisitar novos recursos.
    - Condição de preemptção. Recursos concedidos a um processo não podem ser forçadamente retirados dele.
    - Condição de espera circular. Deve existir um encadeiamento circular de dois ou mais processos.

6.2.2 – Modelagem de impasses

- Pode-se modelar impasses usando grafos.
- Usa-se 4 estratégias para lidar com impasses
    - Ignorar por completo o problema.
    - Detecção e recuperação.
    - Anulação dinâmica por eio da alocação cuidadosa de recursos.
    - Prevenção, negando estruturalmente uma das quatro condições necessárias para gerar um impasse.

6.3 – Algoritmo do avestruz

- Algoritmo de ingnorar o problema. Dependendo da área, da frequecia e do custo de um impasse, pode ser inaceitável

6.4 – Detecção e recupereração de impasses

6.4.1 – Detecção de impasses com um recurso de cada tipo

- Constrói grafo de recursos.
- Se houver um arco, há um impasse.

6.4.2 – Detecção de impasses com múltiplos recursos de cada tipo

- Vetor de recursos existentes.
- Vetor de recursos disponíveis.
- Matriz de alocação atual
- Matriz de requisições.
- Verificar se ocorre impasse assim q entrega um recurso.

6.4.3 – Recuperação de situações de impasse

Recuperação por meio de preempção

- Caso o recurso seja preemptivel, retira-se o recurso. Alguns recursos não preemptiveis podem se tornar preempetiveis sob algumas condições;

Recuperação por meio de retrocesso.

- Cria checkpoints ao longo da execução de um arquivo.

- Se ocorrer um impasse o processo é reiniciaizado num ponto quando ele não tinha o recurso dividido.

Recuperação por meio de eliminação de processos.

- Matar um processo no impasse.
- Matar um processo capaz de ser reiniciado.

6.5 – Evitando impasses

- Pode-se evitar impasses avaliando se a liberação de um recurso é segura ou não.

6.5.1 – Trajetória de recursos.

6.5.2 – Estados seguros e inseguros

- Um estado é considerado seguro e ele não está em situação de impasse e se existe alguma ordem de escalonamento na qual todo processo possa ser executado at´sua conclusão.
- Um estado inseguro não é uma situação de impasse.
- A diferenã entre um estado seguro e inseguro é que a partir de um estado seguro o sistema pode garantir que todos os estados terminarão.

6.5.3 – O algoritmo do banqueiro para um único recurso.

- O algoritmo verficica se a liberação de um recurso pode levar a um estado inseguro. Em caso positivo a requisição desse recurso será negada.

6.5.4 – O algoritmo do banqueiro com múltiplos recursos.

- O algoritmo não é muito útil na prática, pois os processos na maioria das vezes não sabem quais recursos irão alocar. O número de processos não é fixo. Além disso, recursos podem desaparecer repentinamente.

6.6 – Prevenção de impasses

- Garantir que pelo menos uma das quatro condições nunca seja satisfeita.

6.6.1 – Atacando a condição de exclusão mútua.

- Um exemplo é o daemon de impressão. Único processo autorizado a imprimir.
- Evitar alocar um recurso quando ele não é absolutamente necessário e limitar a quatidade de processos que acessam um recurso.

6.6.2 – Atacando a condição de posse e espera

- Impedir que processos com recusos requisitem mais recursos.
- Processo só pode alocar todos os recursos que vai precisar de uma só vez. Causa problemas pois os recursos não serão usados de forma otimizada e muitos processos não sabem os recursos que irão utilizar.
- O recurso que está pedindo outro libera os que estão sob sua posse e requisita todos novamente.

6.6.3 – Atacando a condição de preempção

- Virtualizando certos recursos pode torna-los preemptiveis.

6.6.4 – Atacando a condição de espera circular

- Restringir a posse de recursos a apenas 1. Se um processo quer outro recurso deve primeiro liberar o que está em sua posse.
- Recursos devem ser alocados seguindo uma ordem numérica.
- Não funciona bem par uma quantidade muito grande de recursos.

6.7.4 – Condição de inanição

- Evita-se a condição de inanição usando uma alocação envolvendo FIFO.