

Sistemas Operativos

Deadlocks

3º ano - ESI e IGE (2011/2012)

Engenheiro Anilton Silva Fernandes
(afernandes@ipiaget.net)

Deadlocks

- Ocorre quando um determinado processo fica em estado Waiting (espera) eternamente
 - O recurso esta sendo usado por outros processos em espera.
- Portanto, **Deadlocks** ou impasse,
 - Quando ocorre, automaticamente um conjunto de processos estão nesse estado
 - está-se em estado de deadlock quando todos os processos no conjunto estão esperando por um evento que só pode ser causado por outro processo do mesmo conjunto.

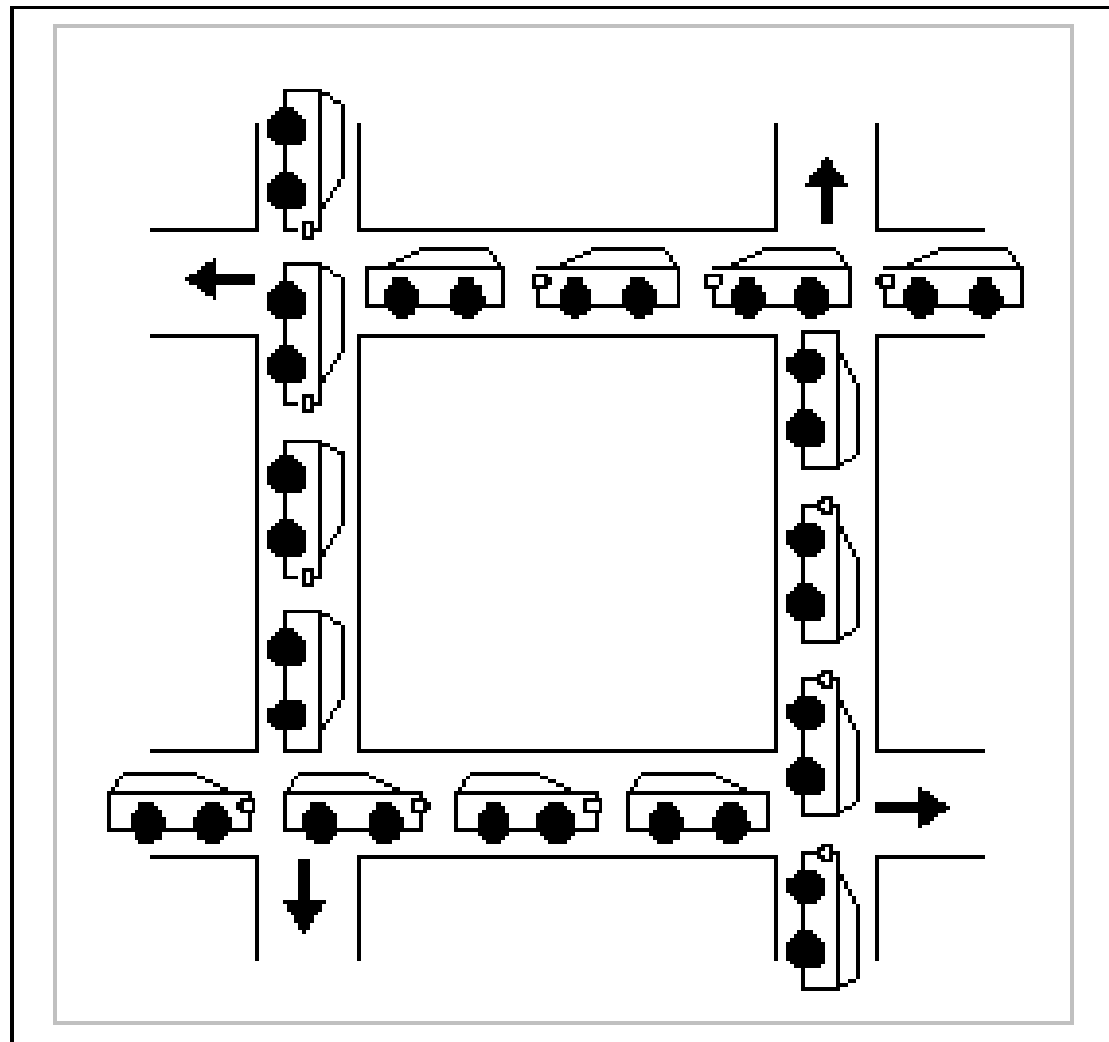
Deadlocks

- Assim,
 - Um deadlock acontece quando duas ou mais tarefas bloqueiam uma à outra permanentemente, sendo que cada uma tem o bloqueio de um recurso, que a outra tarefa está tentando bloquear.
- Por exemplo:
 - A transação A adquire um bloqueio compartilhado da linha 1.
 - A transação B adquire um bloqueio compartilhado da linha 2.
 - A transação A agora solicita um bloqueio exclusivo na linha 2 e é bloqueado até que a transação B termine e libere o bloqueio compartilhado que tem na linha 2.
 - A transação B agora solicita um bloqueio exclusivo na linha 1 e é bloqueado até que a transação A termine e libere o bloqueio compartilhado que tem na linha 1.

Deadlocks

- Alguns dos mais conhecidos Deadlocks são:
 - **requisições de arquivos:** se for permitido aos programas requisitar e bloquear durante sua execução.
 - P1 requisita e obtém arquivo F2
 - P2 requisita e obtém arquivo F1
 - P1 requisita F1 mas está bloqueado
 - P2 requisita F2 mas está bloqueado.
 - **alocação de dispositivos dedicados:** utilização de um grupo de dispositivos dedicados pode gerar deadlocks. Exemplo: um programa que precise de 2 fitas magnéticas para copiar dados de uma para outra.
 - **alocação de múltiplos dispositivos:** vários processos requisitam e bloqueiam diferentes dispositivos dedicados.

Deadlocks



Deadlocks

- Para que situações de Deadlocks aconteçam, 4 condições precisam acontecer em simultâneo
 - **exclusão mútua**: apenas um processo por vez pode usar o recurso.
 - **monopolização de recursos**: acontece no exemplo da escada quando duas pessoas se cruzam em um lance e elas se recusam a retroceder.
 - **não-preempção**: os recursos só podem ser liberados voluntariamente pelo processo que o mantém.
 - **espera circular**: dado um conjunto de processos $\{P1, P2, \dots, Pn\}$ em espera, P1 está esperando por um recurso mantido por P2; P2 está esperando um recurso mantido por P3 e assim sucessivamente, até que Pn esteja esperando por um recurso alocado por P1.

Deadlocks

- Métodos de tratamento do Deadlock:
 - 3 são os métodos que se pode usar para tratar um Deadlock
 - podemos usar um protocolo para garantir que o sistema nunca entre em estado de deadlock: PREVENÇÃO (o Sistema operativo deve eliminar uma das 4 condições para que o deadlock ocorra)
 - podemos permitir que o sistema entre em Deadlock e se recupere: DETECÇÃO e RECUPERAÇÃO.
 - podemos ignorar o problema (Algoritmo do avestruz). Esta opção é utilizada pela maioria dos sistemas operativos, incluindo o UNIX.

Deadlocks – PREVENIR ou EVITAR

- Garantir que o sistema nunca entre em estado de deadlock
 - Fazer **prevenção** baseia-se num conjunto de regras de requisição de recursos que garantem que pelo menos uma das 4 condições necessárias para DEADLOCK não aconteça
 - **Evitar**, por outro lado, requer que seja fornecida ao sistema operativo informação adicional sobre quais recursos um processo irá requisitar e usar durante sua execução.
 - Assim, é possível saber se o processo pode requisitar recurso ou esperar

Deadlocks – PREVENIR ou EVITAR

- Para então prevenir DEADLOCKS precisamos evitar uma das situações onde ela ocorre
 - Negando a Condição “Mutual Exclusion”
 - A condição “*mutual exclusion*” não deve ser negada
 - Negando a Condição “Hold and Wait”
 - Requer que todos os recursos que um processo precise seja requisitado de uma só vez
 - Negando a Condição “No Preemption”
 - É uma estratégia ainda pior do que a anterior
 - Negando a Condição “Circular Wait”
 - ...

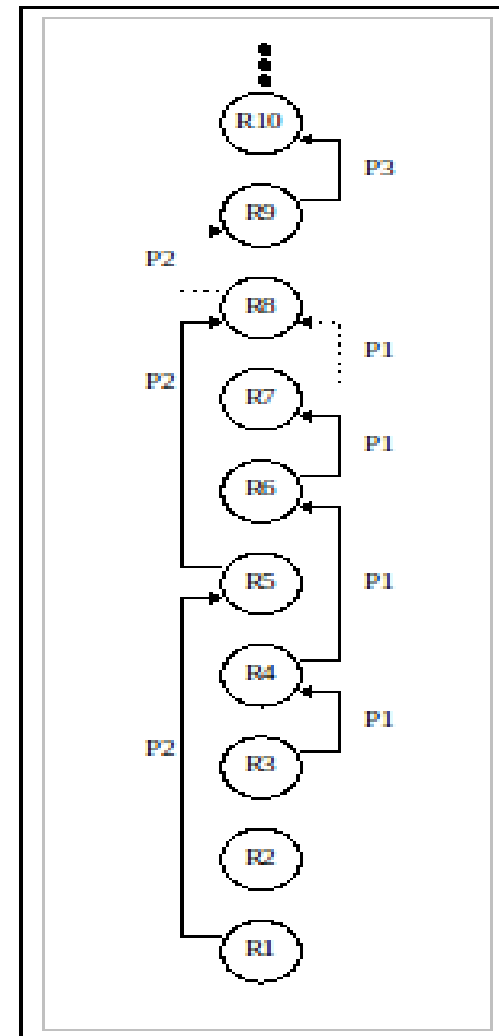
Deadlocks – PREVENIR ou EVITAR

- Negando a Condição “Circular Wait”
 - Podemos ... estabelecer uma regra que diz que um processo só pode alocar um único recurso em um dado momento.
 - Se precisar de outro, tem que desalocar o recurso em seu poder
 - Ou fazer melhor ... utilizar a terceira estratégia de Havender
 - todos os recursos devem ser numerados em ordem crescente
 - Assim, processos podem requisitar recursos sempre que quiserem, mas todas as requisições devem ser em ordem crescente de numeração

Deadlocks – PREVENIR ou EVITAR

- Negando a Condição “Circular Wait”

um processo poderia requisitar o recurso R1 e em seguida o recurso R3, mas não o inverso. O grafo de alocação mostrado na figura 2.7 jamais possuirá ciclos, evitando assim a condição “*circular wait*”.



Deadlocks – DETECÇÃO e RECUPERAÇÃO

- Se não são usadas estratégias de prevenção ou para evitar deadlocks
 - Então podem ser usados algoritmos que consigam ver que ocorreu DEADLOCKS e possivelmente um outro que consiga recuperar a situação
 - Neste caso, estaria-se a fazer a detecção, ou descobrir que ocorreu um DEADLOCK e a fazer o tratamento da situação, desfazendo o impasse

Deadlocks - IGNORAR

- Pode-se também ignorar que situações de DEADLOCKS possam acontecer
 - Quando assim for, não haverá maneira de saber o que aconteceu exatamente.
 - Nesses casos, o DEADLOCK não detectado causará a deterioração do desempenho do sistema, já que recursos estão detidos por processos que não podem continuar, e porque mais e mais processos, conforme requisitam recursos, entram IGUALMENTE em IMPASSE.

Deadlocks – IGNORAR

- Quando a ideia é ignorar, dizemos, que o sistema vai fingir que o problema não existe.
 - É natural que IGNORAR possa parecer uma má ideia, e se os DEADLOCKS forem uma constante, é de certo uma péssima ideia.
 - Pelo que é razoável se os Deadlocks acontecem muito raramente
 - Quando a ideia for IGNORAR, normalmente o sistema irá parar de funcionar, e terá que ser reinicializado manualmente

Deadlocks – IGNORAR

- Devem dizer, então se o sistema vai a baixo, a ideia não pode ser usada!
 - Mas ...
 - Esta ideia é utilizado em vários sistemas operativos. Como normalmente são sistemas muito bons, *deadlocks* ocorrem de forma não freqüente, como por exemplo, uma vez por ano.
 - Este método também é muito menos dispendioso
 - Esse método é por exemplo usado tanto pelo UNIX, quanto pelo MS WINDOWS