Inanição (computação) - Starvation

Origem: Wikipédia, a enciclopédia livre.
Ir para: navegação, pesquisa

Em programação concorrente, ocorre **inanição** quando um processo nunca é executado ("morre de fome"), pois processos de prioridade maior sempre o impedem de ser executado. Em um ambiente computacional multitarefa, a execução de diversos processos simultâneos deve seguir uma regra de escalonamento destes para uso do processador. Isso se deve ao fato de que, durante a mudança de contexto pelo processador, é feita a escolha do próximo processo a ser executado a partir da prioridade deste. Quando o escalonamento não é feito adequadamente, pode haver inanição. Uma solução para esta situação é a delegação de um tempo máximo de espera.

A ocorrência da inanição se dá quando os programas rodam indefinidamente (razão pela qual também se dá o nome de preterição indefinida a esta situação) e não fazem nenhum progresso em seu processamento, ao contrário do *deadlock*, que ocorre quando os processos permanecem bloqueados, dependendo da liberação dos recursos por eles alocados.

Em um sistema dinâmico, as requisições de recursos ocorrem durante todo o tempo. Algumas políticas são necessárias para subsidiar a decisão de quem vai ficar com qual recurso e em que momento. Essas políticas, apesar de parecerem extremamente razoáveis, podem fazer com que alguns processos nunca sejam servidos, apesar de não estarem em *deadlock*.

Um exemplo é a alocação de uma impressora. Considerando que o sistema utilize algum algoritmo que garanta que a alocação da impressora não vai levar a situações de *deadlock*, e considerando também que diversos processos desejam utilizá-la ao mesmo tempo, deve-se definir qual dos processos tem o direito de usá-la.

Um algoritmo possível para implementar a alocação da impressora é o que escolhe o processo com o menor arquivo a ser impresso (assumindo que esta informação está disponível). Este algoritmo maximiza o número de usuários satisfeitos com o sistema, e parece ser um algoritmo justo. Consideremos, no entanto, o que acontece num sistema muito carregado, quando determinado processo tem um arquivo imenso a ser impresso. Cada vez que a impressora estiver disponível, o sistema vai escolher, para usar a impressora, o processo com o menor arquivo a ser impresso. Se houver um fluxo constante de processos no sistema com arquivos pequenos, aquele ou aqueles com arquivos grandes jamais poderão usar a impressora. Eles apenas “morrerão de fome” (como o próprio sentido da palavra inanição estabelece), ou seja, serão preteridos indefinidamente em favor de outros, como se estivessem bloqueados.

A preterição por tempo indeterminado pode ser evitada usando-se uma política de alocação baseada na regra do primeiro-a-chegar é o primeiro-a-ser-servido. Com esta abordagem, o processo que espera há mais tempo é o primeiro a receber serviço por parte do recurso liberado. Fica claro que qualquer dos processos será o mais antigo com o passar do tempo, recebendo, assim, o direito ao uso do recurso pelo qual estiver esperando.

Outra solução é fazer com que o processo em inanição aumente sua prioridade de acordo com o seu tempo de espera, o que também pode resolver o problema.

Exemplos de Inanição

Em processos trocando mensagens: • Dois processos enviando repetitivamente mensagens um para o outro e um terceiro bloqueado, esperando por uma mensagem de um deles. Outra propriedade de um programa paralelo a ser evitada é a injustiça, ou não equanimidade, que é também conhecida por inanição, ou starvation, em inglês; Neste caso não se configura um bloqueio do sistema, mas um processo tem o seu progresso impedido pela ação de outros processos; Nos leitores e escritores, um escritor pode esperar indefinidamente pela autorização para escrever, se por exemplo a demanda por leituras for alta o suficiente para nunca termos número de leitores igual 0. Pode-se mostrar uma computação de um algoritmo paralelo que leve a um deadlock. A inanição só se revela em computações infinitas, e por isto exige uma demonstração formal. A inanição só acontece quando não podemos garantir que o processo será executado.

Problemas comuns

Muitos programadores de sistema operacional tem o conceito de prioridade do processo. Um processo de alta prioridade será executado antes de um processo de baixa prioridade B. Se o processo de alta prioridade nunca bloqueia, o processo de prioridade baixa (B) nunca (em alguns sistemas) se foi agendada - irá experimentar a inanição. Se houver um processo de maior prioridade do que X, que é dependente de um resultado do processo B, então o processo X nunca pode acabar, mesmo que seja o processo mais importante no sistema. Esta condição é chamada de inversão de prioridade. Algoritmos de escalonamento modernos normalmente contêm código para garantir que todos os processos receberão um montante mínimo de cada recurso importante (na maioria das vezes o tempo de CPU), a fim de evitar qualquer processo de ser submetido a inanição.

Em redes de computadores, em especial as redes sem fio, algoritmos de escalonamento podem conter escalonamento de starvation . Um exemplo é a programação de transferência máxima.