1ª. PROVA DE SISTEMAS DE COMPUTAÇÃO II – 2010/2

GABARITO

1ª Questão

Processos I/O-bound apresentam ciclos de CPU de curta duração ao longo de toda a sua execução. Como o algoritmo estabelece que processos que usaram menos a CPU no passado recente são favorecidos, processos I/O-bound são naturalmente beneficiados por esse esquema: devido ao fato de usarem pouco a CPU, a sua prioridade de escalonamento está sempre em patamares elevados.

Ao contrário dos processos I/O-bound, processos CPU-bound possuem longos ciclos de CPU não sendo, portanto, diretamente beneficiados pelo algoritmo. Entretanto, embora não sejam escalonados com freqüência, processos CPU-bound não entram em situação de *starvation* pois se beneficiam da característica de *aging* (envelhecimento) do algoritmo: devido ao seu histórico de pouca posse da CPU no passado recente, a sua prioridade é gradativamente aumentada, tendo com isso garantida a sua execução em algum momento.

Critério de correção:

Entendimento do porquê I/O bound é privilegiado e CPU-bound não é privilegiado : 2,0 pontos

Explicação do porquê processos CPU-bound não entram em starvation (mencionando a característica de *aging*): 2,0 pontos

2ª. Questão:

Ao longo da sua execução, processos fazem uso dos diversos recursos do sistema de computação. Em um S.O. monoprogramado, por exemplo, um processo de usuário não tem que disputá-los com outros processos de usuário. Deste modo, a execução de um processo é mais rápida e com menor tempo de resposta do que teria caso executado em um sistema multiprogramado, onde existe uma disputa natural pelos recursos do sistema entre os processos ativos. Pode-se deduzir que quanto mais recursos um processo detém, maior é a ociosidade dos recursos do sistema de computação, menor é a probabilidade do processo entrar em estado de espera por recursos, e, conseqüentemente, mais rápida tende a ser a sua execução.

Certamente, uma maneira do sistema minimizar o tempo de resposta de um particular processo seria reservar o maior número possível de recursos para ele. Com isso, o sistema garantiria a execução do processo com mínima espera por recursos. Entretanto essa estratégia se contrapõe ao objetivo geral de maximização dos recursos do sistema pois ela é conseguida às custas da ociosidade de alguns desses recursos nos momentos em que o processo não estivessem utilizando-os. Para maximizar a utilização dos recursos, eles deveriam estar distribuídos entre os vários processos, diminuindo a probabilidade de ociosidade que existe quando se aloca muitos desses recursos a um único processo. Conclui-se, assim, que os

objetivos de maximização de uso dos recursos do sistema a minimização dos tempos de resposta dos processos são objetivos conflitantes.

Critério de correção:

Entender que quanto mais recursos detém um processo, menor tempo ele passa em filas de espera por recursos e, conseqüentemente, menor é o seu tempo de resposta (mais rápida é a sua execução): 1,0 ponto

Entendimento de que isso diminui a taxa de utilização dos recursos (por que os objetivos se contrapõem): 1,0 ponto.

3ª. Questão:

Em sistemas Unix, um processo é dito executar em kernel mode (estado kernel running) quando solicita algum serviço do kernel. Em um kernel Unix não-preemptivo, um processo executando em kernel mode não perde a posse do processador mesmo com a chegada de processos mais prioritários na fila de prontos ou mesmo que o seu quantum expire.

Nessa abordagem, as funções e serviços do kernel solicitadas pelos processos de usuário têm a garantia de serem executados até o final, sem "suspensão", já que não é permitida a troca de processos durante a operação em kernel mode. Isso garante maior segurança e integridade na manipulação das estruturas de dados do kernel. Nesse sentido, essa característica faz do kernel não-preemptivo uma solução ampla para a maioria dos problemas de sincronização entre processos.

Por outro lado, essa abordagem introduz o problema geral da inversão de prioridade: processos menos prioritários detendo a posse da CPU em detrimento de processos mais prioritários. Em especial, essa característica faz do kernel não-preemptivo totalmente inadequado para processos de tempo real. Devido a natureza desses processos, de fortes restrições temporais e de uso em sistemas de monitoramento e controle, o tempo de espera a que são submetidos aguardando a conclusão da operação em kernel mode pode ultrapassar os limites temporais permitidos, podendo resultar em danos para o sistema físico sendo controlado.

Critério de correção:

Entendimento do que é um kernel não-preemptivo: 2,0 pontos

Vantagem: 1,0 ponto

Desvantagem: 1,0 ponto