Lucas Alves Fidelis Araújo
Atividade de Avaliação 2

1) O problema da inversão de prioridades discutido em sala na Seção 2.3.4 acontece com threads de usuário? Por que ou por que não? (1.5)

Não, pois as threads de usuário são criados em espaço de usuário somente, o núcleo não tem conhecimento de sua existência, por isso as threads de usuários não sofrem com o problema de inversão de prioridade.

2) Considere a solução a seguir para o problema da exclusão mútua envolvendo dois processos P0 e P1. Presuma que a variável turn seja inicializada para 0. Para o processo P1, substitua 0 por 1 no código anterior. Determine se a solução atende a todas as condições exigidas para uma solução de exclusão mútua (2.0)

Para cumprir todas as condições exigidas para uma solução, seria necessário corrigir o processo P1, pois o mesmo não deixa o P0 acessar a região crítica, mesmo não estando nela. Sendo assim, a solução dada não atende todas as condições exigidas.

3) Explique o funcionamento dos algoritmos escalonamento garantido e por loteria. Por que o escalonamento por loteria é uma versão mais simples de implementar? (2.0)

Escalonamento garantido: Existe um nível de promessa aos usuários. Exemplo: Se houver n usuários, cada um receberá 1/n de CPU. O sistema deve manter a quantidade de CPU que cada processo usou.

Escalonamento por loteria: Ele é mais simples de ser implementado do que o escalonamento garantido, pois os processos ganham "bilhetes". Os processos com mais prioridade ganham bilhetes, e cada processo podem trocar bilhetes entre si. Um sorteio é realizado e quem tiver o bilhete, pode executar. Dessa forma, não fica a cargo do SO coordenar o tempo dos processos, até mesmo em nível de usuário se poderia gerir essa troca de bilhetes.

4) Cinco processos A, B, C, D e E chegam nesta ordem, conforme tempos apontados, com os seguintes CPU burst e prioridades (maior valor significa maior prioridade).

	CPU Burst	Prioridade	Chegada
A	3	3	1
B	7	5	0
C	4	1	5
D	1	4	2
E	2	2	6

Simule a execução dos algoritmos FCFS, job mais curto primeiro, prioridades com 5 classes e RR (em cada classe, quantum =1) e round-robin (quantum=2). Considere que tempo de troca de contexto é desprezível. Para cada algoritmo simulado apresente: o tempo de espera de cada processo e o tempo médio de espera. (3.0)

5) Julgue as sentenças a seguir como Verdadeiras ou Falsas. Justifique suas respostas para as sentenças avaliadas como Falsas. (1.5)

a. **VERDADEIRO** Uma das quatro condições para uma boa solução de IPC é não assumir algo sobre a velocidade de execução dos processos/threads ou o número de CPUs disponível.

b. **FALSO** O problema do jantar dos filósofos é um problema tradicional de IPC. Na versão tradicional deste problema temos 5 filósofos sentados em uma mesa buscando comer o prato de macarrão a sua frente fazendo uso do garfo à sua esquerda. O problema consiste em coordenar a velocidade de comida dos filósofos de modo que todos acabem ao mesmo tempo.

Cada filósofo alterna entre duas tarefas: comer ou pensar. Quando um filósofo fica com fome, ele tenta pegar os garfos à sua esquerda e à sua direita; um de cada vez, independente da ordem. Caso ele consiga pegar dois garfos, ele come durante um determinado tempo e depois recoloca os garfos na mesa. Em seguida ele volta a pensar.

O problema do jantar dos filósofos é representativo de uma grande classe de problemas de sincronização entre vários processos e vários recursos sem usar um coordenador central. Resolver o problema do jantar dos filósofos consiste em encontrar uma forma de coordenar suas atividades de maneira que todos os filósofos consigam meditar e comer.

c. **FALSO** Processos sempre tem um mesmo comportamento: usam a CPU por longos períodos de tempo e em seguida fazem alguma requisição de E/S.

De acordo com seu comportamento de execução de instruções processos podem ser classificados:

Orientado à CPU (cpu-bound): maior parte do tempo no estado de execução ou pronto

Orientado à E/S (io-bound): maior parte do tempo no estado de espera (alto número de operações de E/S)

d. **VERDADEIRO** Escalonabilidade é o conceito relacionado com a ideia de conseguirmos verificar se é possível escalonar os processos que estão chegando em um sistema de tempo real e, assim, atender aos requisitos do sistema.

e. **VERDADEIRO** Um algoritmo de escalonamento preemptivo é aquele que considera a marcação do relógio para controlar por quanto tempo um processo executa. Já um algoritmo não-preemptivo executa o processo até que ele seja bloqueado ou libere a CPU.