Avaliação 02

1. Sobre comunicação interprocessos, defina: (1.5)

- **a. Condição de corrida (disputa)**

Condição em que dois processos lêem e escrevem um dado compartilhado e o resultado final depende da ordem em que os processos são executados.
Uma solução para o problema da condição de corrida é evitar que mais de um processo leia/escreva ao mesmo tempo. Essa estratégia é chamada exclusão mútua (mutual exclusion) mutex.

- **b. Região Crítica**

A região crítica de um determinado processo é a parte de seu código que acessa a Área de **Memória Compartilhada**, que, como o próprio nome sugere, é um conjunto de recursos que podem ser compartilhados entre dois ou mais processos, por exemplo, memória, variáveis, arquivos, etc. Essa área depende expressivamente que a parte do processo que a acessa tenha sua execução de forma sequencial, e esse é um dos principais fatores que influenciam para a ocorrência de uma condição de corrida.

- **c. Exclusão Mútua**

A exclusão mútua (mutex) é um objeto de programa que impede o acesso simultâneo a um recurso compartilhado. Este conceito é usado em programação concorrente com uma seção crítica, um pedaço de código que processa ou tópicos acessar um **recurso compartilhado.** Apenas um segmento possui o mutex de cada vez, assim, um mutex com um nome exclusivo é criado quando um programa é iniciado. Quando um segmento detém um recurso, tem que bloquear o mutex de outros tópicos para impedir o acesso simultâneo do recurso. Após a liberação do recurso, a thread libera o mutex.

- **d. Recurso compartilhado** (Mostrado em negrito em tópico anteriores).

Possibilidade de leitura e escrita por vários processos (ou threads).

2. É possível determinar se um processo é propenso a se tornar limitado pela CPU ou limitado pela E/S analisando o código fonte? Como isso pode ser determinado no tempo de execução? (2.0)

Através do código fonte é possível verificar a frequência com a qual operações E/S são executadas, já durante a execução deve-se utilizar ferramentas que medem a porcentagem de uso da CPU por um dado programa. Entretanto, isto somente é válido em máquinas que permitem apenas um único usuário.

3. O que é problema de inversão de prioridade discutido em sala? O problema da inversão de prioridades acontece com threads de usuário? Por que ou por que não? (2.0)

O problema de inversão de prioridades seria quando uma tarefa de alta prioridade é impedida de executar por causa de uma tarefa de baixa prioridade.

Pode ocorrer se:

- A tarefa de alta prioridade precisa de um recurso.
- Esse recurso está com a tarefa de baixa prioridade.
- A CPU está ocupada com outras tarefas.
- A tarefa de baixa prioridade não consegue executar.

Não, pois as threads de usuário são criadas 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.

4. Explique 3 objetivos de algoritmos de escalonamento especificando se este objetivo é um objetivo global ou mais voltado para algum tipo de sistema. (1.0)

Voltado para todos os sistemas:

- Justiça – atribuir porções justas de CPU a processos similares
- Aplicação da política – Verificar se a política estabelecida é cumprida
- Equilíbrio – Manter todo o sistema ocupado balanceando E/S e CPU

5. Diferencie o funcionamento dos algoritmos próximo de menor tempo restante e job mais curto primeiro. (1.5)

O algoritmo de job mais curto primeiro (Shortest Job First) consiste em atribuir o processador à menor (mais curta) tarefa da fila de tarefas prontas.
Uma versão preemptiva da tarefa mais curta primeiro é o tempo restante mais curto em seguida (shortest remaining time next). Com esse algoritmo, o escalonador escolhe o processo cujo tempo de execução restante é o mais curto. De novo, o tempo de execução precisa ser antecipado. Quando uma nova tarefa chega, seu tempo total é comparado com o tempo restante do processo atual. Se a nova tarefa precisa de menos tempo para terminar do que o processo atual, este é suspenso e a nova tarefa iniciada. Esse esquema permite que tarefas curtas novas tenham um bom desempenho.

6. Cinco tarefas em lote, A até E, chegam a um centro de computadores quase ao mesmo tempo. Elas têm tempos de execução estimados de 10, 6, 2, 4 e 8 minutos. Suas prioridades (externamente determinadas) são 3, 5, 2, 1 e 4, respectivamente, sendo 5 a mais alta. Para cada um dos algoritmos de escalonamento a seguir, determine o tempo de retorno médio do processo. Ignore a sobrecarga de chaveamento de processo. (2.0)

a. Circular
b. Escalonamento por prioridade
c. Primeiro a chegar, primeiro a ser servido (siga a ordem 10, 6, 2, 4, 8)

Para (a), presuma que o sistema é multiprogramado e que cada tarefa recebe sua porção justa de tempo na CPU. Para (b) e (c), presuma que apenas uma tarefa de cada

vez é executada, até terminar. Todas as tarefas são completamente limitadas pela CPU.

a) Tempo: 2

A	2					2				2			2		2	=	10
B		2					2				2					=	6
C			2													=	2
D				2				2								=	4
E					2				2			2		2		=	8

Tempo médio = 6

b) (4+2+10+8+6) /5 = 6
c) (10+6+2+4+8) /5 = 6