1. Quais recursos são utilizados quando uma thread é criada? Como eles diferem daqueles usados quando um processo é criado?

2. Explique com suas próprias palavras o conceito de multiprogramação em sistemas operacionais.

3. Sobre processos explique: O que é um processo? Como funciona a criação de um processo?

4. Explique a diferença entre um escalonador preemptivo e um não-preemptivo. A frase "Um escalonador preemptivo apresenta sempre melhor desempenho que um escalonador não-preemptivo" é verdadeira ou falsa? Justifique a sua resposta.

5. Considere 4 processos (A, B, C, D) com os seguintes tempos de CPU: 8, 5, 6, e 7 segundos respectivamente. Todos são processos *CPU-bound* (não fazem I/O), de mesma prioridade e escalonados segundo a política *Round-Robin* com um *quantum* de 5 segundos. Todos os processos chegam ao sistema (i.e., são disparados) respectivamente em 0, 4, 9 e 14 segundos. Calcule o tempo médio de espera e o tempo médio de resposta do lote de processos.

6. Considere 4 processos (A, B, C, D) com os seguintes tempos de CPU: 2, 8, 3, e 5 segundos respectivamente. Todos são processos *CPU-bound* (não fazem I/O), e possuem as seguintes prioridades: A 3, B 1, C 2, D 3. Esses processos são escalonados segundo a política *de* ***prioridade estática***. Todos os processos chegam ao sistema (i.e., são disparados) respectivamente em 0, 1, 2 e 3 segundos. Calcule o tempo médio de espera e o tempo médio de resposta do lote de processos, considerando o escalonamento preemptivo e o não preemptivo.

7. Explique com suas próprias palavras o método de escalonamento First-Come-First-Served (FCFS), e o escalonamento Shortest-Job-First (SJF). Na sua opinião qual a desvantagem do SJF não preemtivo para o preemptivo?

8. Quais recursos são utilizados quando uma thread é criada? Como eles diferem daqueles usados quando um processo é criado?

9. Quais são as condições necessárias para garantir a execução coordenada de uma seção crítica? Explique cada uma delas.

10. Com suas próprias palavras explique o que são seções críticas.

11. Considerando os seguintes trechos de código do problema produtor-consumidor:

Produtor

```
1: shared int counter = 0;
2: shared char buf[N];
3:
4:
5: int main()
6: {
7:       int in = 0;
8:
9:       while (true) {
10:
11:        while (counter == N);
12:
13:        buf[in] = produce();
14:
15:        in = ++in % N;
16:
17:        counter++;
18:      }
19: }
```

Consumidor

```
1: shared int counter = 0;
2: shared char buf[N];
3:
4:
5: int main()
6: {
7:       int in = 0;
8:
9:       while (true) {
10:
11:        while (counter == 0);
12:
13:        Consume(buf[out]);
14:
15:        out = ++out % N;
16:
17:        counter--;
18:      }
19: }
```

a) Identifique as seções críticas;

b) Faça a proposta de uma solução para exclusão mútua;

c) Explique o motivo da solução escolhida;

d) Insira as primitivas de sincronização de sua solução no código.

12. Marque verdadeiro ou falso nas afirmativas abaixo. Corrija as erradas.

( ) O desempenho da política de escalonamento *Round-Robin* é dependente do tamanho do *quantum* utilizado.

( ) Um *processo* compartilha código, dados e pilha com as demais processos.

( ) A política de escalonamento First-Come-First-Served (FCFS), além de muito eficiente garante que um processo sempre estará livre de inanição.