Sistemas Operacionais

Primeiro Estágio

1. **O que são interrupções e exceções?**

2. **Explique o que é um processo? Descreva as estruturas de dados que compõem um e as responsabilidades de cada uma.**

3. **Por que no código abaixo não ocorrem condições de corrida.**


```python
import os
import sys
import threading

def do_sum(path):
    _sum = 0
    try:
        with open(path, 'rb', buffering=0) as f:
            byte = f.read(1)
            while byte:
                _sum += int.from_bytes(byte, byteorder='big', signed=False)
                byte = f.read(1)
    except Exception as e:
        print(f"Error processing {path}: {e}")
    finally:
        print(f"{path} : {_sum}")

def process_file(path):
    _sum = do_sum(path)

if __name__ == "__main__":
    paths = sys.argv[1:]
    threads = []

    for path in paths:
        t = threading.Thread(target=process_file, args=(path,))
        t.start()

```

4. **Como e com quais finalidades o SO utiliza os níveis de processamento (modos de execução)?**

5. **Diga quais são e explique os estados e as transições entre processos.**
    a. **Desenhe um diagrama de estados/transições, contendo os 3 principais estados pelos quais um processo pode passar.**
    b. **Explique os eventos que podem causar essas transições e como o sistema operacional ganha o controle da CPU para realizá-las.**

6. **Como e por que ocorrem condições de corrida e como evitar?**

7. **Como o SO pode executar uma operação down de forma atômica e por que ser atômica?**

8. **Uma vez que um processo do usuário está de posse da CPU, em que condições o sistema operacional pode voltar a executar?**

9. **Suponha um programa que executa as threads D, T1, T2 e T3. D é a Thread despachante, ela é responsável pela recepção de serviços e colocação do pedido num dos três buffers apropriados para aquele serviço. T1, T2 e T3 são responsáveis pela realização de serviços distintos. Cada uma delas tem um buffer associado. Escreva um pseudocódigo usando semáforos de D, T1, T2 e T3 considerando que cada buffer tem capacidade para armazenar no máximo N pedidos. Modifique sua solução para o caso em que todas as threads compartilham um mesmo buffer.**

11. **Explique como um escalonador de processos Round-Robin é implementado. Descreva o algoritmo a ser usado, explicitando as estruturas de dados usadas, bem como em que instantes cada parte do algoritmo é executada.**

12. **Explique em quais condições possam ocorrer impasses (DEADLOCKS).**

13. **Considere duas threads, A e B, que precisam executar dois métodos dependentes entre si, garanta que a chamada do método bar_2() só ocorra após foo(), assim como, executar bar() apenas após foo_2(). Crie um pseudocódigo que lide com essa sincronização.**

Thread A	Thread B
fn foo();	fn foo_2();
fn bar();	fn bar_2();

14. **Usando semáforos, escreva o pseudocódigo de um tipo abstrato de estrutura de dados, cujas instâncias podem ser compartilhadas por várias threads,que tem como estado um vetor de 10 posições para armazenar inteiros e que implementa as seguintes operações:**
    a. **void put (int dado): esse método tenta armazenar o parâmetro dadona estrutura de dados que armazena o estado da instância em qualquer posição livre do vetor; se não há espaço para armazenar mais um inteiro, a chamada bloqueia;**
    b. **int get (dado); esse método retorna algum inteiro armazenado na estrutura de dados; se não há nenhum dado armazenado, a chamada bloqueia;**
    c. **int sumOdd(); esse método retorna a soma dos inteiros armazenados nas posições ímpares do vetor, ou zero se não há nenhum inteiro armazenado nessas posições; e**

d. **int sumEven(); esse método retorna a soma dos inteiros armazenados nas posições pares do vetor, ou zero se não há nenhum inteiro armazenado nessas posições.**

15. **Dado o código abaixo, onde além dos semáforos vazio, cheio, e muter, é também compartilhado um buffer com um número finito de posições, onde as informações produzidas são armazenadas e de onde as informações a serem consumidas são retiradas:**

```
semaphore vazio = X;
semaphore cheio = Y;
mutex = Z;
produtor() {
        while (VERDADE) {
                produz_item();
                down(&vazio);
                down(&mutex);
                adiciona_item();
                up(&mutex);
                up(&cheio);
        }
}
consumidor() {
        while(VERDADE) {
                down(&cheio);
                down(&mutex);
                remove_item();
                up(&mutex);
                up(&vazio);
                imprime_item();
        }
}
```

Indique os valores de X (vazio), Y (cheio) e Z (mutex) para atender às seguintes especificações do problema do produtor consumidor:

a. **Existem 10 posições no buffer, que inicialmente então ocupadas, um produtor e um consumidor.**

b. **Existem 20 posições no buffer, que inicialmente estão vagas, um produtor e um consumidor.**

c. **Existem 15 posições no buffer, das quais, inicialmente, 10 posições estão ocupadas e 5 estão livres, dois produtores e três consumidores.**