1. Descreva como dois processos podem se comunicar, usando memória compartilhada.

Criação da Memória Compartilhada: Um processo cria uma região de memória que será compartilhada e outros processos podem se conectar a essa região.

Acesso à Memória: Uma vez que a memória compartilhada é estabelecida, os processos podem ler ou escrever diretamente na memória. O acesso simultâneo à memória compartilhada pode levar a problemas de consistência, então mecanismos de sincronização são geralmente necessários.

2. O que é uma condição de corrida?

Condição de Corrida ocorre quando dois ou mais processos ou threads tentam acessar e modificar uma mesma variável ou recurso compartilhado ao mesmo tempo.

3. Descreva uma situação em que pode haver uma condição de corrida.

Dois processos tentando incrementar o valor de uma mesma variável compartilhada:

Processo 1 lê o valor da variável (digamos, 5).

Processo 2 lê o valor da mesma variável (também 5).

Ambos os processos incrementam o valor lido para 6.

Processo 1 escreve 6 na variável.

Processo 2 também escreve 6 na variável.

Apesar de dois incrementos terem sido feitos, o valor final da variável é 6, em vez de 7, devido à condição de corrida.

4. O que é uma região crítica de um processo? Por que ela precisa ser protegida de execução simultânea?

Parte do código onde o acesso a recursos compartilhados (como variáveis ou memória) acontece. Para evitar condições de corrida, é crucial que apenas um processo execute a região crítica por vez. Se mais de um processo entrar na região crítica simultaneamente, pode haver inconsistências nos dados.

Proteger a região crítica é essencial para evitar que múltiplos processos alterem os dados simultaneamente, o que pode levar a inconsistências e erros imprevisíveis. A proteção pode ser feita usando técnicas como mutexes, semáforos, ou monitores.

6. O que é espera ocupada?

Espera Ocupada é uma técnica onde um processo fica em um loop verificando continuamente se uma condição foi atendida (por exemplo, se um recurso está disponível). Durante a espera, o processo consome tempo de CPU sem fazer nenhum trabalho útil, o que pode ser ineficiente.

7. **Descreva as seguintes estratégias de comunicação/sincronização entre processos (Monitores; Semáforos; Mutex; Test Set and Lock (variável de lock)).**

Monitores: Monitores são estruturas de alto nível que encapsulam variáveis, procedimentos e a região crítica. Eles garantem que apenas um processo execute dentro da região crítica do monitor por vez, simplificando a sincronização.

Semáforos: Um semáforo é uma variável de sincronização que pode controlar o acesso a recursos compartilhados por múltiplos processos. Existem dois tipos básicos de semáforos: semáforos binários (semáforos que podem ser 0 ou 1) e semáforos de contagem (que podem ter valores maiores que 1). Semáforos são usados para evitar condições de corrida e para coordenar a execução de processos.

Mutex: Um mutex (mutual exclusion) é semelhante a um semáforo binário, mas é especificamente usado para garantir que apenas um processo ou thread acesse um recurso ou execute uma parte crítica do código por vez. Mutexes geralmente têm operações de bloqueio (lock) e desbloqueio (unlock).

Test Set and Lock (TSL): TSL é uma instrução atômica usada para implementar locks. Ela verifica o valor de uma variável de lock e, se o recurso estiver disponível (por exemplo, o valor da variável é 0), o recurso é bloqueado (mudando o valor para 1) e o processo entra na região crítica. Se o valor já for 1, o processo continua esperando ou entra em espera ocupada.