Universidad Nacional Autónoma de México

Facultad de Ciencias

Computación Concurrente

Practica 02

Alumnos:
David Pérez Jacome
Valeria Fernanda Manjarrez Angeles

Profesor: Jorge Luis Ortega Arjona

Ayudante: Gibran Aguilar Zuñiga

Abril 2023

PRACTICA 02

Preguntas:

Deberán detallar a profundidad las respuestas y en caso de ser necesario hacer diagramas para ejemplificar tu respuesta.

1. ¿Qué es un proceso? (1 punto).

   Es el cambio en el estado de la memoria por acción del procesador.

2. ¿Qué es la sección crítica de un proceso? (1 punto).

   Parte del codigo donde se encuentran y modifican los recursos compartidos.

3. ¿Qué es el problema libre de hambruna (2 puntos).

   Se refiere a que tenemos que garantizar que los procesos no se queden sin uso de procesador o de recursos que necesite.

4. ¿Qué es el problema de abrazos mortales? (2 puntos).

   Es un problema matematico que implica el número de formas en la que se pueden dar la mano varias personas en una reunión sin que existan dos aprentones de manos al mismo tiempo. La solución al problema es que el número total de apretones de manos será igual a $\frac{(n(n-1))}{2}$. Esto se debe a que cada persona puede dar la mano a $n-1$ personas diferentes (ya que no puede darse la mano consigo misma), lo que significa que el número total de apretones de manos es igual a la suma de todos los apretones de manos individuales, que es igual a $n(n-1)$. Como cada apretón de manos se cuenta dos veces (una vez por cada persona que la realiza), se divide el resultado por 2 para obtener el número total de apretones de manos únicos.

5. Haz un TDA de Hilos y un TDA de Semáforos (4 puntos).

   Para semaforos:

```
        / Declaración del TDA de semáforos
TDA_Semaforo:

    // Atributos
    valor: entero
    cola_espera: cola de procesos

    // Métodos wait() y signal()
    crear(valor_inicial: entero) -> semáforo:
        semáforo = nueva instancia de TDA_Semaforo
        semáforo.valor = valor_inicial
        semáforo.cola_espera = nueva cola vacía
```

```
        retornar semáforo

    wait(semáforo: semáforo):
        semáforo.valor = semáforo.valor - 1
        si semáforo.valor < 0 entonces:
            agregar proceso actual a cola_espera
            suspender proceso actual

    signal(semáforo: semáforo):
        semáforo.valor = semáforo.valor + 1
        si semáforo.valor <= 0 entonces:
            sacar proceso de cola_espera
            despertar proceso
```

Para Hilos:

```
        // Declaración del TDA de hilos
TDA_Hilo:

    // Atributos
    id: entero
    estado: enum {ejecutando, listo, bloqueado}
    contexto: contexto del hilo

    // Métodos
    crear(funcion: funcion, argumentos: lista) -> hilo:
        hilo = nueva instancia de TDA_Hilo
        hilo.id = generar_id_unico()
        hilo.estado = listo
        hilo.contexto = crear_contexto(funcion, argumentos)
        retornar hilo

    correr(hilo: hilo):
        cambiar_estado(hilo, ejecutando)
        guardar_contexto(hilo.contexto)
        llamar_funcion(hilo.contexto.funcion, hilo.contexto.argumentos)
        cambiar_estado(hilo, terminado)

    cambiar_estado(hilo: hilo, nuevo_estado: enum):
        hilo.estado = nuevo_estado
```

Ejecución:

Para ejecutar nuestro programa es necesario:

1. Localizarnos en en directorio: **Practica02**.
2. Ejecutamos: **mvn compile**
3. Ejecutamos el ejecutable que se localiza en el directorio target: **java -jar target/Practica01.jar**.