PRÁCTICA

PROBLEMA DE LOS FILÓSOFOS COMENSALES

Objetivos

- **Objetivo General:**

Comprender y aplicar los conceptos de creación, concurrencia y sincronización en C para resolver problemas de computación paralela. Aplicando semáforos para manejar la concurrencia de los hilos y evitar interbloqueos.

- **Objetivos específicos:**
    - Analizar y comprender el funcionamiento y la implementación de hilos con phtread_create().
    - Diseñar procesos concurrentes.
    - Implementar mecanismos de sincronización mediante semáforos para controlar el acceso a recursos compartidos y establecer un orden específico en la ejecución de hilos.

Requisitos

- Lenguaje C instalado.
- IDE o Editor de Texto a elección.
- Sistema Operativo Linux con distro a elección.

Descripción del Problema

Cinco filósofos se sientan alrededor de una mesa circular. Cada filósofo alterna entre pensar y comer. Para comer espaguetis, un filósofo necesita exactamente dos tenedores: el de su izquierda y el de su derecha. Solo hay cinco tenedores disponibles en total.

Se requiere que implemente dos soluciones:

1. Una versión ingenua donde cada filósofo toma primero el tenedor izquierdo y luego el derecho.
2. Una versión que utiliza un semáforo contador para limitar el número máximo de filósofos que pueden intentar comer simultáneamente.

Requerimientos Tecnicos

- Lenguaje C con pthread library
- Semáforos POSIX
- Función usleep() para simular tiempos de pensamiento y comida.

Estructura Base

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
#include <time.h>

#define NUM_FILOSOFOS 5
#define TIEMPO_PENSAR_MIN 1000000
#define TIEMPO_PENSAR_MAX 3000000
#define TIEMPO_COMER_MIN 500000
#define TIEMPO_COMER_MAX 1500000
#define TIEMPO_SIMULACION 30

// Variables globales
sem_t tenedores[NUM_FILOSOFOS];
sem_t comedor;
pthread_mutex_t mutex_print;
int filosofos_comiendo[NUM_FILOSOFOS];
int total_comidas[NUM_FILOSOFOS];
```

Escenarios de Prueba

Escenario 1: Detectar cuando ocurre un interbloqueo.

Configuración:

- 5 Filosofos
- Tiempos de pensamiento cortos de menos de 1 segundo.
- Tiempos de comida largos (2 - 3 segundos)
- Ejecutar por al menos 10 segundos.

Objetivos: Demuestra que la primera solución llega rápidamente a un interbloqueo y que la segunda solución funciona correctamente.

Evidencia: Adjuntar captura de pantalla de cuando ocurre el interbloqueo y cuando la solución funciona correctamente. **Nota:** Agregar prints necesarios para evidenciar sus resultados.

Escenario 2: Análisis de la equidad.

Configuración:

- 5 Filosofos
- Tiempos balanceados (pensar: 1-2s, comer: 0.5-1s)
- Ejecutar por 60 segundos

Objetivo: Verificar que la solución con semáforos no causa inanición. Deberá contar el número de comidas por cada filósofo.

Evidencia: Adjuntar captura de pantalla de la cantidad de comida por cada filósofo.

Escenario 3: Análisis de robustez.

Configuración:

- 3 Filósofos
- Tiempos muy cortos (pensar: 50ms - 100ms, comer: 100ms - 300ms)
- Ejecutar por 30 segundos

Objetivo: Verificar que la solución con semáforos funcione correctamente con un alto nivel de concurrencia

Evidencia: Adjuntar captura de pantalla del programa ejecutado correctamente después de 30 segundos, y el número de comidas por filósofo.

Escenario 4: Escenarios variables.

Configuración:

- Probar la solución con 2, 7, 12 filósofos
- Ajustar el semáforo a N-1 filósofos.
- Tiempos estándar.
- Ejecutar por 20 segundos cada escenario.

Objetivo: Probar cómo se desarrolla la solución con distintas cantidades de filósofos.

Evidencia: Adjuntar captura de pantalla del programa ejecutado correctamente en cada escenario.

Análisis de Resultados

1. ¿En qué momento se produce el interbloqueo en la primera versión?
2. ¿Cómo afecta el número de filósofos al rendimiento de la solución?
3. En alguno de los escenarios se produjo inanición? ¿Por qué?
4. ¿Cómo se comportaría el sistema si un filósofo tiene tiempos de comida mucho mayores que los demás?

Entregables

1. Código Fuente

- Implementación con interbloqueo
- Implementación sin interbloqueo

2. Reporte de Análisis

En el documento de reporte debe incluir:

- Explicación del interbloqueo observado
- Análisis de la efectividad de la solución planteada.
- Gráficos de estadísticas por escenario.
- Capturas de cada escenario.
- Respuestas a las preguntas de análisis.

Penalizaciones

- Uso de Inteligencia Artificial: -60%.
- Entrega fuera de tiempo: -40%.
- No se adjuntó evidencia de la solución: -50%.
- No se utilizó el lenguaje C: -85%.