Laboratorio 08: sincronización con semáforos

Instrucciones.

Esta actividad se realizará individualmente. Al finalizar los períodos de laboratorio o clase, deberá dejar constancia de sus avances en Canvas, según indique su catedrático. Al finalizar la actividad, adjuntar los archivos .pdf y .cpp para solucionar los ejercicios:

- Desarrolle el programa solución en C++, empleando Pthreads, semáforos y/o barreras.
- Incluir video corto con explicación de funcionamiento del programa.

Ejercicio 01

24 puntos: 4 pts cada inciso. Crea un cuadro comparativo para ayudar a entender las diferencias y similitudes entre varios mecanismos de sincronización y control de hilos en la programación concurrente. Los conceptos a comparar son:

- Mutex (Exclusión Mutua)
- Variables de Condición
- Semáforos
- Barreras

Toma en cuenta los siguientes aspectos antes de generar el cuadro comparativo:
Identificar las características clave a comparar:

1. **Función principal:** cuál es el propósito o la función principal de cada mecanismo.
2. **Cuándo se usa:** explica en qué situaciones es más adecuado utilizar cada mecanismo.
3. **Sincronización:** ¿se utiliza para sincronización de hilos o para exclusión mutua?
4. **Número de hilos involucrados:** ¿cuántos hilos pueden participar en la operación?
5. **Bloqueo/Espera:** ¿el mecanismo implica que uno o más hilos esperen/bloqueen la ejecución hasta que se cumpla una condición o hasta que otros hilos terminen?
6. **Reinicialización:** ¿Es posible o necesario reinicializar el mecanismo después de su uso?

Aspecto	Mutex (Exclusión Mutua)	Variables de Condición	Semáforos	Barreras
Función principal	Garantizar que solo un hilo tenga acceso a una sección crítica a la vez (exclusión mutua).	Coordinar la ejecución de hilos basado en condiciones (esperar hasta que se cumpla una condición).	Controlar el acceso a recursos compartidos y contar/limitar los hilos que pueden realizar ciertas operaciones.	Sincronizar un grupo de hilos para que todos alcancen un punto de ejecución antes de continuar.
Cuándo se usa	Cuando se necesita evitar condiciones de carrera asegurando que solo un hilo acceda a un recurso o sección de código al mismo tiempo.	Cuando un hilo debe esperar a que una condición específica sea verdadera antes de proceder (ej. productor-consumidor).	Para limitar el número de hilos que pueden acceder a un recurso compartido o realizar una acción simultáneamente.	Cuando se necesita que un conjunto de hilos avance de manera sincronizada, esperando a que todos lleguen a un cierto punto.
Sincronización	Exclusión mutua para proteger recursos.	Sincronización de hilos en función de una condición.	Exclusión mutua y sincronización de hilos.	Sincronización de hilos, todos deben alcanzar la barrera antes de continuar.
Número de hilos involucrados	Usualmente entre 2 o más hilos.	Usualmente entre 2 o más hilos, pero puede variar según la condición.	Puede usarse con cualquier número de hilos.	Múltiples hilos (debe ser igual al número configurado para la barrera).
Bloqueo/Espera	El hilo que no obtiene el mutex es bloqueado hasta que el recurso esté disponible.	Los hilos se bloquean si la condición no se cumple, hasta que otro hilo la señale.	Los hilos pueden bloquearse si el semáforo está en cero o si no hay "permiso" disponible.	Todos los hilos se bloquean hasta que el último hilo llega a la barrera.
Reinicialización	No es necesario reinicializarlo manualmente; se libera cuando el hilo termina.	La variable de condición no se reinicia automáticamente; debe ser señalada explícitamente por un hilo.	Puede ser reinicializado automáticamente cuando se incrementa el valor del semáforo.	Se reinicia automáticamente después de que todos los hilos han alcanzado la barrera.

Ejercicio 02

32 puntos: En una red peer-to-peer P2P (https://www.obsbusiness.school/blog/que-son-las-redes-p2p-y-cual-su-utilidad), varios nodos intentan compartir archivos. Cada nodo actúa como cliente y servidor, solicitando archivos de otros nodos y compartiendo los archivos que tienen. Usaremos hilos para representar a los nodos en la red, y semáforos para controlar el acceso a los archivos compartidos, asegurando que solo un nodo a la vez pueda acceder o modificar la lista de archivos disponibles.

Requisitos:

- **5 pts. Nodos**: Cada nodo es representado por un hilo que participa en la red P2P.
- **5 pts. Archivos Compartidos**: Los nodos comparten un conjunto de archivos en común y pueden solicitar archivos de otros nodos.
- **5 pts. Semáforos**: Se utilizan semáforos para proteger el acceso a la lista de archivos compartidos y evitar condiciones de carrera cuando varios nodos intentan acceder al mismo archivo.
- **5 pts. Operaciones**:
  1. **Descargar un archivo**: Un nodo puede solicitar un archivo de otro nodo. Si el archivo está disponible, se simula la descarga.
  2. **Subir un archivo**: Los nodos también pueden compartir archivos que descargaron, añadiéndolos a la lista de archivos compartidos.

```
Ingrese el numero de nodos en la red P2P: 5
Nodo 2 intenta descargar archivo3.txt
Nodo 2 ha descargado archivo3.txt
Nodo 3 intenta descargar archivo3.txt
Nodo 3 ha descargado archivo3.txt
Nodo 1 intenta descargar archivo2.txt
Nodo 1 ha descargado archivo2.txt
Nodo 4 intenta descargar archivo3.txt
Nodo 4 ha descargado archivo3.txt
Nodo 5 intenta descargar archivo3.txt
Nodo 5 ha descargado archivo3.txt
```

Responde:

1. **3 pts.** ¿Cuál es el recurso compartido que debe protegerse de posibles condiciones de carrera?
   El recurso compartido que debe protegerse es la lista de archivos compartidos. Dado que varios hilos (nodos) pueden intentar leer, modificar o añadir elementos a esta lista al mismo tiempo, es crucial protegerla para evitar condiciones de carrera.

2. **3 pts.** ¿Qué pasa si no se utiliza ninguna forma de sincronización al acceder al recurso compartido?
   Sin ninguna forma de sincronización, se producirían race conditions, lo que significa que varios hilos podrían acceder y modificar la lista de archivos compartidos simultáneamente. Esto podría llevar a situaciones en las que: Varios nodos intenten descargar el mismo archivo a la vez, lo que genera inconsistencias y un nodo intente añadir un nuevo archivo mientras otro está descargando, causando errores de acceso, corrupción de datos, o resultados inesperados.

3. **3 pts.** ¿Cómo sabrás que un archivo ya haya sido descargado?
   En el código, se usa el atributo disponible en cada archivo para indicar su estado. Si el archivo tiene *disponible = false*, significa que ya está siendo descargado por otro nodo, evitando que otros intenten descargarlo simultáneamente.

4. **3 pts.** Si utilizas semáforo, ¿cuál es el valor inicial del semáforo y por qué?
   El semáforo se inicializa con un valor de 1. Esto garantiza que solo un hilo a la vez pueda acceder a la lista de archivos compartidos. Un valor inicial de 1 en el semáforo implementa una exclusión mutua, evitando que múltiples hilos ingresen a la sección crítica donde se gestiona la lista de archivos compartidos.

Ejercicio 03

34 puntos. Una fábrica está compuesta por dos tipos de empleados, donde **productores** fabrican piezas de **sillas** y las colocan en un almacén de tamaño limitado. A su vez, los **consumidores** son ensambladores que retiran las piezas del almacén para ensamblar las sillas. Para fabricar una silla completa, se deben utilizar 1 respaldo, 1 asiento y 4 patas.
Los hilos representan tanto a los productores como a los consumidores, y usamos semáforos para controlar el acceso concurrente al almacén, asegurándonos de que no haya sobreproducción cuando el almacén está lleno, ni intentos de retirar piezas cuando no hay productos disponibles.

Analiza el programa Ejercicio03A.cpp para comprender la simulación generada a través de código.

Indica y resuelve. Responde (deja evidencia de código):

10 pts. Explica el funcionamiento del sistema ¿Se inician y finalizan correctamente los proceso? ¿Por qué?
El código está diseñado para que los hilos de los productores y consumidores continúen ejecutándose indefinidamente, lo que significa que el sistema no finaliza por sí mismo una vez que se produce un número determinado de sillas. Por lo tanto, los procesos de producción y consumo no terminan automáticamente, y esto debe modificarse para limitar la producción. El sistema utiliza varios componentes clave:

Productores: Cada productor selecciona una pieza de silla al azar (pata, respaldo o asiento) y la añade al buffer, simulando la fabricación de una pieza.
Consumidores: Cada consumidor retira piezas del buffer, y cuando tiene las suficientes (1 respaldo, 1 asiento, y 4 patas), ensambla una silla.

Sincronización:

Semáforos vacios para controlar el espacio en el buffer, que permite a los productores añadir piezas solo si hay espacio disponible.
Semáforos llenos para asegurar que los consumidores retiren piezas solo si hay alguna disponible.
Mutex protege el acceso al buffer, evitando condiciones de carrera entre hilos.

1. **10 pts.** ¿Qué modificaciones deben realizarse para que los productores dejen de producir cuando se alcance el límite de sillas, y los consumidores también terminen?

   Para que los productores dejen de producir cuando se alcance un límite, por ejemplo, MAX_SILLAS, y que los consumidores también terminen, se puede hacer lo siguiente:
   - Crear una variable global que cuente cuántas sillas se han ensamblado y compararla con el MAX_SILLAS.
   - Incluir una condición en el ciclo while de los productores y consumidores que verifique si ya se alcanzó el límite de sillas, de modo que terminen su ejecución cuando este límite se cumple.
   - Usar una condición adicional para evitar que los consumidores sigan esperando piezas cuando ya no habrá más producción, permitiendo que ambos procesos finalicen juntos.

2. **14 pts.** Agrega el código necesario para que se genere un reporte antes de finalizar la ejecución. El reporte debe indicar cuántas sillas se fabricaron en totalidad, y cuántas piezas de cada tipo sobraron (quedarán en almacén).