PROBLEMAS DE COMUNICACIÓN MÁS COMUNES ENTRE PROCESOS

Problema de la cena de los filósofos / problema de los cinco filósofos comensales.

" Cinco filósofos se sientan a la mesa. Cada uno tiene un plato de espagueti. El espagueti es tan escurridizo, que un filósofo necesita dos tenedores para comerlo. Entre cada dos platos hay un tenedor."

Este problema hace referencia a la complejidad de organizar procesos que están en competencia por el acceso exclusivo a un número limitado de recursos. En este caso los filósofos compitiendo por los tenedores.

Supongamos aquí que los filósofos solo tienen tres estados básicos: comer, detenerse a pensar, o esperar a que otro comensal termine. Si cualquier filósofo toma un tenedor y el otro está ocupado, se quedará en espera con el tenedor en la mano, hasta que pueda tomar el otro tenedor, para luego empezar a comer esperar (es decir el filósofo que este comiendo bloqueará al de su lado). Por lo tanto, el resto de los filósofos que no tiene ni un tenedor están pensando.

Como son 5 filósofos lo más efectivo sería permitir que la mayor parte del tiempo haya dos filósofos comiendo. Pero debemos evitar casos como el interbloqueo o deadlock: si por ejemplo, todos toman el tenedor que está a su derecha, entonces se quedarán esperando eternamente, todos estarán en modo de espera.

Unas soluciones más efectivas que encontré y que consiste en mantener casi todo el tiempo a dos filósofos comiendo y evitando completamente los bloqueos, es la siguiente: "cada filósofo debe acceder al comedor consiguiendo un turno de los cuatro que tiene el portero del comedor, después debe coger un tenedor y si en un tiempo aleatorio consigue el otro tenedor come, pero si no lo consigue debe soltar el tenedor que tiene, abandonar el comedor y salir a pensar."

Problema de los lectores y los escritores.

" Hay un objeto de datos (fichero de texto) que es utilizado por varios procesos, unos leen y otro que escribe. Solo puede utilizar el recurso un proceso y solo uno, es decir, o bien un proceso estará escribiendo o bien leyendo, pero nunca ocurrirá simultáneamente"

Este problema responde a la necesidad de organización del acceso a una base de datos.

Las condiciones del problema son las siguientes:

- Cualquier número de lectores pueden leer el archivo simultáneamente.
- Solo puede escribir en el archivo un escritor a la vez.
- Si hay escritor accediendo al archivo, ningún lector podrá leerlo.

- El escritor tendrá prioridad sobre el lector.

Una posible solución es el cambiar esta prioridad, haciendo que sea el escritor el que tenga que esperar a que no haya escritores para poder editar, para así no dejar esperando a los varios lectores que quieran acceder al archivo.

Problema del barbero dormilón.

Una peluquería tiene un barbero, una silla de peluquero y varias sillas para que se sienten los clientes en espera. Si no hay clientes, el barbero se sienta en la silla de peluquero y se duerme, es así que cuando llega un cliente debe despertar al barbero.

Un cliente que entra a la peluquería debe contar el número de clientes que esperan, si este es menor que el número de sillas, se queda; en caso contrario se va.

Una solución que encontré, un tanto compleja propone la implementación de un semáforo de la siguiente forma:

- Cuando el barbero abre su negocio se debe ejecutar un semáforo denominado *barber* que checa el número de barberos en espera de clientes (0 o 1), lo que establece un bloqueo en otro semáforo: *customer*, que cuenta el número de clientes en espera, después se va a dormir.
- Cuando llega el primer cliente, éste ejecuta *customer*, que inicia procurando que un tercer semáforo llamado *mutex* entre en una región crítica. *Mutex* se va a utilizar para la exclusión mutua.
- Si otro cliente llega, no podrá hacer nada hasta que el primero haya liberado a *mutex*.
- El cliente verifica entonces si el número de clientes que esperan es menor que el número de sillas.
- Si esto no ocurre, libera a *mutex* y sale sin su corte de pelo. • Si existe una silla disponible, el cliente incrementa la variable entera *waiting*, que es una réplica de *customer*. Después realiza un “levantamiento” en *customer*, con lo que despierta al barbero.
- Cuando el cliente libera a *mutex*, el barbero lo retiene, ordena algunas cosas e inicia el corte de pelo.
- Al terminar el corte, el cliente sale del procedimiento y deja la peluquería.

Problema del productor - consumidor.

Este es uno de los problemas que surgen al utilizar procesos concurrentes, es decir aquellos donde existen procesos llamados productores los cuales generan datos que son consumidos por otros procesos llamados consumidores.

Una solución implica la implementación de los siguientes 3 semáforos par:

Mutex: accede al buffer y sus variables.

Vacío: cuenta el número de elementos vacíos (inicializado a N).

Lleno: cuenta el número de elementos (inicializado a 0).

De tal manera que:

- El proceso consumidor no puede extraer datos si el buffer está vacío.
- El proceso productor no puede introducir datos si el buffer está lleno.
- El semáforo Mutex controla la entrada y salida de la Sección Crítica, compuesta por el buffer limitado, los apuntadores de entrada y salida al mismo y el contador del número de elementos.

Linkografía:

https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiChKaPmJX9AhU8hu4BHdQkBzEQFnoECAsQAQ&url=https%3A%2F%2Facademicos.azc.uam.mx%2Fjestrada%2Farchivos%2Fso%2Ftema4%2Fp_clasicos.pdf&usg=AOvVaw3a7h_dnxyHnEDPRUY7x8Fn

https://www.infor.uva.es/~cllamas/concurr/pract98/sisos30/index.html#:~:text=La%20solución%20de%20este%20problema,memoria%20compartida%20(véase%20"rshmem.

https://www.calameo.com/read/00330823117c5c319cfe1

https://pacoportillo.es/informatica-avanzada/programacion-multiproceso/la-cena-de-los-filosofos/

https://www.infor.uva.es/~cllamas/concurr/pract97/rsantos/index.html