PROBLEMAS CLÁSICOS DE LA COMUNICACIÓN ENTRE PROCESOS

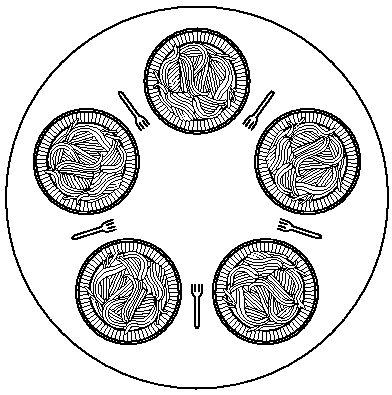
SISTEMAS OPERATIVOS
JOSÉ ALFREDO ESTRADA SOTO

Para cada uno de los siguientes tres problemas se requiere escribir un programa que atienda las necesidades que presenta cada uno de estos, aplicando la solución a la administración de procesos que presenta un sistema operativo.

EL PROBLEMA DE LA CENA DE LOS FILÓSÓFOS

- Este problema es útil para modelar procesos que están en competencia por el acceso exclusivo a un número limitado de recursos

- Este problema se puede enunciar de la siguiente manera: 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.



- La vida de un filósofo consta de dos periodos alternados de comer y pensar. (Esto es una abstracción, incluso para los filósofos, pero las demás actividades son irrelevantes en este caso).
- Cuando un filósofo tiene hambre, intenta obtener un tenedor para su mano izquierda y otro para su mano derecha, alcanzando uno a la vez y en cualquier orden.
- Si logra obtener los dos tenedores, come un rato y después deja los tenedores y continúa pensando.

- Una solución es esperar hasta que el tenedor especificado este disponible y tomarlo. Esta solución es incorrecta si se hace la suposición de que los cinco filósofos toman sus tenedores izquierdos en forma simultánea. Ninguno de ellos podría tomar su tenedor derecho, con lo que ocurriría un bloqueo.
- Al volver a colocar los tenedores en la mesa, ninguno estaría comiendo por lo que intentarían volver a tomarlos y esta situación se repetiría indefinidamente y es lo que se denomina como **inanición**.

- Otra solución es esperar un cierto tiempo arbitrario, en lugar del mismo tiempo, después de que no pudiesen tomar el tenedor derecho y así reducir a un mínimo la probabilidad de inanición pero hay ciertas aplicaciones críticas que requieren funcionalidad total que esta solución no proporciona.

- Una mejora de las soluciones anteriores es utilizar un semáforo para indicar que recurso se tiene en uso. Esto solo permite tener a un filósofo con dos tenedores.

- Si un filósofo requiere comer debe verificar si a los lados no se están utilizando los dos tenedores, obteniendo así la solución más viable para este problema.

EL PROBLEMA DE LOS LECTORES Y LOS ESCRITORES

Se utiliza para modelar el acceso a una base de datos.

- Cuando se tiene una enorme base de datos, como por ejemplo un sistema de reservaciones de una aerolínea, con muchos procesos en disputa por la escritura o lectura a dicha base. ¿Cómo se deben programar los lectores y los escritores?.

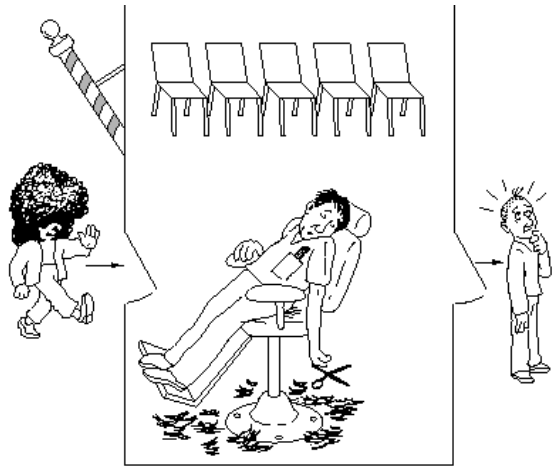
- Una solución es dar en primera instancia prioridad primaria a los lectores y secundaria a los escritores, es decir, si existen lectores el escritor debe esperar hasta que ya no haya lectores.

- En segunda instancia, cuando el escritor tome prioridad primaria los lectores deben esperar a que el escritor termine su función.

EL PROBLEMA DEL BARBERO DORMILÓN

El problema consiste en programar a un barbero y a sus clientes sin entrar en competencia

- La peluquería tiene un barbero, una silla de peluquero y *n* sillas para que se sienten los clientes en espera, si es que los hay.



- Si no hay clientes el barbero se sienta en la silla de peluquero y se duerme. Cuando llega un cliente debe despertar al barbero.

- Si llegan más clientes mientras el barbero corta el cabello a un cliente, ellos se sientan (si hay sillas desocupadas).

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

Una solución

- 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 replica 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.

- Nótese que no existe un ciclo para el cliente, puesto que el corte de pelo es una actividad que el cliente recibe automáticamente, más no ejecuta tal actividad.