**5.1. Enumere cuatro aspectos de diseño para los cuales el concepto de concurrencia es relevante.**

**Concurrencia:** los procesos se entrelazan en el tiempo para ofrecer la apariencia de ejecución simultánea. Aunque no se consigue procesamiento paralelo real, e ir cambiando de un proceso a otro supone cierta sobrecarga, la ejecución entrelazada proporciona importantes beneficios en la eficiencia del procesamiento y en la estructuración de los programas. En un sistema de múltiples procesadores no sólo es posible entrelazar la ejecución de múltiples procesos sino también solaparlas.

**¿Qué aspectos de diseño y gestión surgen por la existencia de la concurrencia?**

Pueden enumerarse las siguientes necesidades:

1. El sistema operativo debe ser capaz de seguir la pista de varios procesos. Esto se consigue con el uso de bloques de control de proceso.
2. El sistema operativo debe ubicar y desubicar varios recursos para cada proceso activo. Estos recursos incluyen:
   * **Tiempo de procesador.** Esta es la misión de la planificación, tratada en la Parte Cuatro.
   * **Memoria.** La mayoría de los sistemas operativos usan un esquema de memoria virtual. El tema es abordado en la Parte Tres.
   * **Ficheros.** Tratados en el Capítulo 12.
   * **Dispositivos de E/S.** Tratados en el Capítulo 11.
3. El sistema operativo debe proteger los datos y recursos físicos de cada proceso frente a interferencias involuntarias de otros procesos. Esto involucra técnicas que relacionan memoria, ficheros y dispositivos de E/S. En el Capítulo 15 se encuentra tratado en general el tema de la protección.
4. El funcionamiento de un proceso y el resultado que produzca, debe ser independiente de la velocidad a la que suceda su ejecución en relación con la velocidad de otros procesos concurrentes. Este es el tema de este capítulo.

**5.2. ¿En qué tres contextos aparece la concurrencia?**

La concurrencia aparece en tres contextos diferentes:

**• Múltiples aplicaciones.** La multiprogramación fue ideada para permitir compartir dinámicamente el tiempo de procesamiento entre varias aplicaciones activas.

**• Aplicaciones estructuradas.** Como extensión de los principios del diseño modular y de la programación estructurada, algunas aplicaciones pueden ser programadas eficazmente como un conjunto de procesos concurrentes.

**• Estructura del sistema operativo.** Las mismas ventajas constructivas son aplicables a la programación de sistemas y, de hecho, los sistemas operativos son a menudo implementados en sí mismos como un conjunto de procesos o hilos.

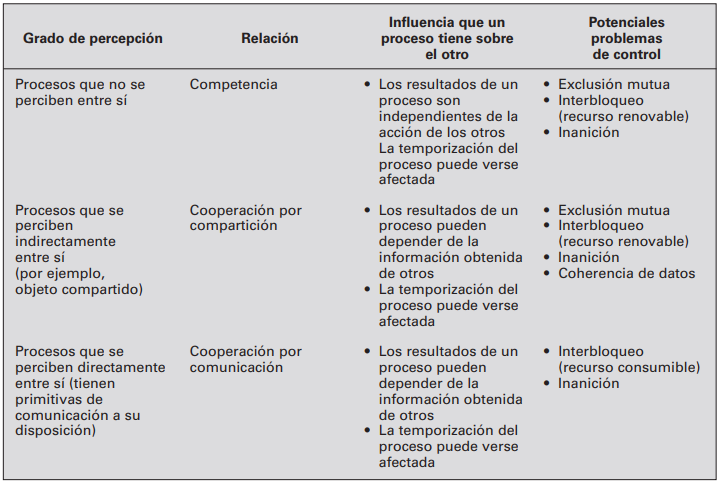
**5.3. ¿Cuál es el requisito básico para la ejecución de procesos concurrentes?**

Se descubre que el requisito básico para conseguir ofrecer procesos concurrentes es la capacidad de hacer imperar la exclusión mutua; esto es, la capacidad de impedir a cualquier proceso realizar una acción mientras se le haya permitido a otro.

**5.4. Enumere tres grados de percepción entre procesos y defina brevemente cada uno.**

Podemos clasificar las formas en que los procesos interaccionan en base al grado en que perciben la existencia de cada uno de los otros. Tres posibles grados de percepción más las consecuencias de cada uno:

* **Procesos que no se perciben entre sí.** Son procesos independientes que no se pretende que trabajen juntos. El mejor ejemplo de esta situación es la multiprogramación de múltiples procesos independientes. Estos bien pueden ser trabajos por lotes o bien sesiones interactivas o una mezcla. Aunque los procesos no estén trabajando juntos, el sistema operativo necesita preocuparse de la **competencia** por recursos. Por ejemplo, dos aplicaciones independientes pueden querer ambas acceder al mismo disco, fichero o impresora. El sistema operativo debe regular estos accesos.
* **Procesos que se perciben indirectamente entre sí.** Son procesos que no están necesariamente al tanto de la presencia de los demás mediante sus respectivos ID de proceso, pero que comparten accesos a algún objeto, como un *buffer* de E/S. Tales procesos exhiben **cooperación** en la compartición del objeto común.
* **Procesos que se perciben directamente entre sí.** Son procesos capaces de comunicarse entre sí vía el ID del proceso y que son diseñados para trabajar conjuntamente en cierta actividad. De nuevo, tales procesos exhiben **cooperación**.



**5.5 ¿Cuál es la diferencia entre procesos en competencia y procesos cooperantes?**

La diferencia principal entre procesos en competencia y procesos cooperantes radica en la naturaleza de su interacción y la forma en que comparten recursos.  
 Mientras que los procesos en competencia compiten por recursos sin tener conocimiento de los demás procesos, los procesos cooperantes pueden compartir recursos de manera compartida o a través de la comunicación directa para coordinar actividades.

**1. Competencia entre procesos por recursos:**

- En este caso, los procesos concurrentes compiten por el uso del mismo recurso.

- Cada proceso opera de manera independiente y no tiene conocimiento explícito de la existencia de otros procesos.

- Los procesos compiten por recursos como dispositivos de E/S, memoria, tiempo de procesador, etc.

- Cada proceso debe dejar inalterado el estado de los recursos que utiliza, y no debe verse afectado por la ejecución de otros procesos.

**2. Cooperación entre procesos vía compartición:**

- En este caso, los procesos interactúan entre sí al compartir recursos como variables, ficheros o bases de datos compartidas.

- Aunque los procesos pueden tener acceso a los mismos datos compartidos, pueden operar de manera independiente sin tener conocimiento explícito de los otros procesos.

- Los procesos deben cooperar para garantizar la integridad de los datos compartidos.

- Los mecanismos de control se utilizan para asegurar que los datos compartidos se manipulen adecuadamente y se mantenga su integridad.

**3. Cooperación entre procesos vía comunicación:**

- En este caso, los procesos participan en un esfuerzo común que los vincula a todos ellos a través de la comunicación directa.

- La comunicación proporciona una manera de sincronizar o coordinar actividades entre los procesos.

- A diferencia de la cooperación vía compartición, donde los procesos no tienen conocimiento explícito de los demás procesos pero son conscientes de la necesidad de mantener la integridad de los datos, en la cooperación vía comunicación, los procesos están directamente involucrados en la coordinación de actividades.

**5.6. Enumere los tres problemas de control asociados con los procesos en competencia y defina brevemente cada uno.**

Los tres problemas de control asociados con los procesos en competencia son:

**1. Exclusión Mutua:** Este problema se refiere a la necesidad de garantizar que solo un proceso tenga acceso a un recurso crítico en un momento dado. Es esencial que, durante la ejecución de una sección crítica de un proceso, ningún otro proceso pueda acceder al mismo recurso para evitar condiciones de carrera y garantizar la consistencia de los datos. La exclusión mutua asegura que los procesos no interfieran entre sí mientras acceden al recurso compartido.

**2. Interbloqueo (Deadlock):** El interbloqueo ocurre cuando dos o más procesos quedan atrapados en un estado de espera perpetua porque cada uno de ellos está esperando que el otro libere un recurso que necesita para avanzar. Por ejemplo, si dos procesos necesitan acceder a dos recursos en el orden opuesto, pueden bloquearse mutuamente mientras esperan que el otro libere el recurso que necesitan. El interbloqueo impide el progreso de los procesos y puede causar una parálisis en el sistema.

**3. Inanición (Starvation):** La inanición ocurre cuando un proceso queda permanentemente excluido de acceder a un recurso compartido, a pesar de que intenta acceder a él repetidamente. Esto puede suceder si otros procesos tienen prioridad sobre el proceso inaniciado y continúan obteniendo acceso al recurso antes de que el proceso inaniciado pueda hacerlo. La inanición puede llevar a una degradación del rendimiento del sistema y a la injusticia en la asignación de recursos.

Estos problemas de control son críticos en la gestión de procesos concurrentes y requieren estrategias y mecanismos adecuados para abordarlos y evitar que ocurran en el sistema. La exclusión mutua es fundamental para garantizar la coherencia de los datos, mientras que el interbloqueo y la inanición deben evitarse para garantizar un funcionamiento adecuado y equitativo del sistema.

**5.7. Enumere las condiciones necesarias para la exclusión mutua.**

Cualquier mecanismo o técnica que vaya a proporcionar exclusión mutua debería cumplimentar los siguientes requisitos:

1. La exclusión mutua debe hacerse cumplir: sólo se permite un proceso al tiempo dentro de su sección crítica, de entre todos los procesos que tienen secciones críticas para el mismo recurso u objeto compartido.
2. Un proceso que se pare en su sección no crítica debe hacerlo sin interferir con otros procesos.
3. No debe ser posible que un proceso que solicite acceso a una sección crítica sea postergado indefinidamente: ni interbloqueo ni inanición.
4. Cuando ningún proceso esté en una sección crítica, a cualquier proceso que solicite entrar en su sección crítica debe permitírsele entrar sin demora.
5. No se hacen suposiciones sobre las velocidades relativas de los procesos ni sobre el número de procesadores.
6. Un proceso permanece dentro de su sección crítica sólo por un tiempo finito.

**5.8. ¿Qué operaciones pueden ser realizadas sobre un semáforo?**

**Definición:** Los *semáforos* son una herramienta de programación utilizada en sistemas operativos y lenguajes de programación para proporcionar concurrencia y sincronización entre procesos o hilos. Fueron introducidos por Edsger Dijkstra en 1965 como una solución para problemas de exclusión mutua y sincronización en sistemas concurrentes.

El concepto fundamental detrás de los semáforos es permitir que los procesos o hilos cooperen mediante señales simples. Un proceso puede ser detenido en un punto específico hasta que reciba una señal específica. Los semáforos son implementados como variables especiales que pueden tener un valor entero no negativo.

Para conseguir el efecto deseado, el semáforo puede ser visto como una variable que contiene un valor entero sobre el cual sólo están definidas tres operaciones:

1. Un semáforo puede ser inicializado a un valor no negativo.
2. La operación semWait decrementa el valor del semáforo. Si el valor pasa a ser negativo, entonces el proceso que está ejecutando semWait se bloquea. En otro caso, el proceso continúa su ejecución.
3. La operación semSignal incrementa el valor del semáforo. Si el valor es menor o igual que cero, entonces se desbloquea uno de los procesos bloqueados en la operación semWait. Aparte de estas tres operaciones no hay manera de inspeccionar o manipular un semáforo.

**5.9. ¿Cuál es la diferencia entre semáforos binarios y semáforos generales? 🡪 No entendí bien**

La diferencia principal entre semáforos binarios y semáforos generales radica en su comportamiento y en la cantidad de valores que pueden tomar.

**1. Semáforos No-binarios: semáforo con contador o** **semáforo general** **:**

- Pueden tomar valores enteros no negativos.

- La operación `semWait` decrementa el valor del semáforo. Si el valor resultante es negativo, el proceso que ejecuta `semWait` se bloquea.

- La operación `semSignal` incrementa el valor del semáforo. Si el valor es menor o igual a cero, se desbloquea uno de los procesos bloqueados en `semWait`.

**2. Semáforos Binarios o Mutex:**

- Solo pueden tomar los valores 0 y 1.

- La operación `semWaitB` (o `semWait` en algunos contextos) comprueba el valor del semáforo. Si el valor es cero, el proceso que ejecuta `semWaitB` se bloquea. Si el valor es uno, se cambia el valor a cero y el proceso continúa su ejecución.

- La operación `semSignalB` (o `semSignal` en algunos contextos) comprueba si hay algún proceso bloqueado en el semáforo. Si hay procesos bloqueados, se desbloquea uno de ellos. Si no hay procesos bloqueados, el valor del semáforo se establece en uno.

**5.10. ¿Cuál es la diferencia entre semáforos fuertes y semáforos débiles?**

La diferencia entre semáforos fuertes y semáforos débiles radica en la política de ordenación de los procesos que están bloqueados y esperan acceder al recurso controlado por el semáforo.

En los semáforos fuertes, se garantiza que los procesos se desbloquearán en el orden en que se bloquearon, lo que puede ser útil en situaciones donde se necesita un comportamiento predecible o justo. Por otro lado, los semáforos débiles no garantizan un orden específico de desbloqueo y dependen del planificador del sistema operativo para determinar qué proceso se desbloquea en qué momento.

**1. Semáforos Fuertes:**

- Utilizan una política FIFO (primero en entrar, primero en salir) para la ordenación de los procesos en la cola de espera.

- En un semáforo fuerte, el proceso que ha estado esperando más tiempo es el primero en ser desbloqueado cuando se libera el recurso.

**2. Semáforos Débiles:**

- No especifican un orden específico para la extracción de procesos de la cola de espera.

- En un semáforo débil, el sistema operativo o el planificador de procesos puede decidir el orden de desbloqueo de los procesos basándose en alguna política de planificación predeterminada.

**5.11. ¿Qué es un monitor? 🡪 Chequear**

Un monitor es un módulo software consistente en uno o más procedimientos, una secuencia de inicialización y datos locales. Las principales características de un monitor son las siguientes: controla sucesos de un programa

1. Las variables locales de datos son sólo accesibles por los procedimientos del monitor y no por ningún procedimiento externo.
2. Un proceso entra en el monitor invocando uno de sus procedimientos.
3. Sólo un proceso puede estar ejecutando dentro del monitor al tiempo; cualquier otro proceso que haya invocado al monitor se bloquea, en espera de que el monitor quede disponible.

**5.12. ¿Cuál es la diferencia entre bloqueante y no bloqueante con respecto a los mensajes?**

La diferencia entre bloqueante y no bloqueante con respecto a los mensajes radica en cómo se comportan los procesos que envían y reciben mensajes:

**- Envío bloqueante, recepción bloqueante:** Tanto el proceso emisor como el receptor se bloquean hasta que el mensaje se entrega. Esta combinación, conocida como rendezvous, asegura que tanto el proceso emisor como el receptor estén sincronizados y esperen activamente hasta que la comunicación se complete.

**- Envío no bloqueante, recepción bloqueante:** El proceso emisor puede continuar su ejecución después de enviar el mensaje, pero el receptor se bloquea hasta que recibe el mensaje solicitado. Esto permite que el proceso emisor avance sin esperar la confirmación de que el mensaje ha sido recibido, mientras que el receptor se bloquea hasta que la comunicación se complete.

**- Envío no bloqueante, recepción no bloqueante:** En esta combinación, ni el proceso emisor ni el receptor se bloquean. Ambos pueden continuar ejecutándose sin esperar la confirmación de la entrega o recepción del mensaje. Esto proporciona la máxima flexibilidad, pero también requiere que los procesos implementen mecanismos para manejar la confirmación de la recepción del mensaje.

Cada combinación tiene sus propias *ventajas y desventajas*. Por ejemplo, el envío no bloqueante y la recepción bloqueante pueden ser útiles para evitar bloqueos completos de procesos mientras se asegura que la comunicación se complete de manera sincronizada. Sin embargo, el envío no bloqueante puede llevar a la generación excesiva de mensajes si no se maneja adecuadamente, lo que puede consumir recursos del sistema.

**5.13. ¿Qué condiciones están asociadas generalmente con el problema lectores/escritores?**

El problema lectores/escritores se define como sigue: Hay un área de datos compartida entre un número de procesos. El área de datos puede ser un fichero, un bloque de memoria principal o incluso un banco de registros del procesador. Hay un número de procesos que sólo leen del área de datos (lectores) y otro número que sólo escriben en el área de datos (escritores). Las siguientes condiciones deben satisfacerse.

1. Cualquier número de lectores pueden leer del fichero simultáneamente.

2. Sólo un escritor al tiempo puede escribir en el fichero.

3. Si un escritor está escribiendo en el fichero ningún lector puede leerlo.