Cuestionario guía - Clases Teóricas 1 y 2

1- **Mencione al menos 3 ejemplos donde pueda encontrarse concurrencia**

- Navegador Web accediendo una página mientras atiende al usuario.
- Varios navegadores accediendo a la misma página.
- Acceso a disco mientras otras aplicaciones siguen funcionando
- Los sistemas biológicos

2- **Escriba una definición de concurrencia. Diferencie procesamiento secuencial, concurrente y paralelo.**

La concurrencia es la capacidad de ejecutar múltiples actividades en paralelo o simultáneamente. Permite a distintos objetos actuar al mismo tiempo

El procesamiento secuencial implica realizar una tarea o proceso a la vez, siguiendo un orden temporal estricto. Cada tarea debe esperar a que la anterior termine antes de comenzar su ejecución.

El procesamiento concurrente se refiere a la ejecución de múltiples tareas o procesos de manera aparentemente simultánea, pero no necesariamente en paralelo. Puede llevarse a cabo en sistemas de un solo procesador mediante la alternancia rápida entre tareas o en sistemas con múltiples procesadores. No está restringido a una arquitectura particular de hardware ni a un número determinado de procesadores. Especificar la concurrencia implica especificar los procesos concurrentes, su comunicación y su sincronización.

El procesamiento paralelo es la ejecución concurrente en múltiples procesadores con el objetivo principal de reducir el tiempo de ejecución.

3- **Describa el concepto de deadlock y qué condiciones deben darse para que ocurra.**

Un deadlock (bloqueo) es una situación en la que dos o más procesos o hilos en un sistema concurrente quedan atrapados en un estado en el que ninguno puede continuar su ejecución debido a que cada uno está esperando que el otro libere un recurso que necesita. Esto puede resultar en un estancamiento completo del sistema.

Condiciones para que ocurra:

1. Recursos reusables serialmente: los procesos comparten recursos que pueden usar con exclusión mutua.
2. Adquisición incremental: los procesos mantienen los recursos que poseen mientras esperar adquirir recursos adicionales.
3. No-preemption: una vez que son adquiridos por un proceso, los recursos no pueden quitarse de manera forzada sino que sólo son liberados voluntariamente.
4. Espera cíclica: existe una cadena circular (ciclo) de procesos tal que cada uno tiene un recurso que su sucesor en el ciclo está esperando adquirir.

Con evitar que se cumpla alguna de las condiciones mencionas, se evita el deadlock. La ausencia de deadlock es una propiedad necesaria en los procesos concurrentes.

4- Defina inanición. Ejemplifique.

La inanición es una situación en la que un proceso o hilo en un sistema concurrente no puede avanzar o realizar su trabajo debido a la incapacidad de acceder a los recursos compartidos o a las condiciones necesarias para su ejecución. Esto puede deberse a que otros procesos tienen prioridad sobre él y constantemente obtienen acceso a los recursos, dejando al proceso en estado de espera indefinida.

Ejemplo:

En un sistema operativo con múltiples procesos compitiendo por el acceso a una impresora compartida. Cada proceso necesita imprimir un documento en la impresora.Si se implementa una política de asignación de recursos que siempre otorga acceso a la impresora al mismo grupo de procesos o al proceso de mayor prioridad, otros procesoscon menor prioridad pueden quedar en estado de inanición.

Si tenemos tres procesos: A, B y C, donde A y B tienen prioridad alta y C tiene prioridad baja, y la política de asignación de recursos da preferencia a los procesos de alta prioridad, entonces el proceso C puede quedarse esperando indefinidamente para imprimir su documento.

5- ¿Qué entiende por no determinismo? ¿Cómo se aplica este concepto a la ejecución concurrente?

En el no determinismo no hay un orden preestablecido en la ejecución, la ejecución de la misma “entrada” puede generar diferentes “salidas”.

Cuando se aplica este concepto a la ejecución concurrente, significa que en un entorno con múltiples hilos o procesos en ejecución, el orden en que se ejecutan las operaciones o eventos no está preestablecido y puede variar de una ejecución a otra, incluso si las entradas y las condiciones iniciales son las mismas.

6- Defina comunicación. Explique los mecanismos de comunicación que conozca.

La comunicación entre procesos concurrentes indica el modo en que se organizan y trasmiten datos entre tareas concurrentes. Esta organización requiere especificar protocolos para controlar el progreso y la corrección.
Los procesos se comunican:

- Por Memoria Compartida.
    - Los procesos intercambian información sobre la memoria compartida o actúan coordinadamente sobre datos residentes en ella.
    - Lógicamente no pueden operar simultáneamente sobre la memoria compartida, lo que obliga a bloquear y liberar el acceso a la memoria.
    - La solución más elemental es una variable de control tipo “semáforo” que habilite o no el acceso de un proceso a la memoria compartida.

- Por Pasaje de Mensajes
    - Es necesario establecer un canal (lógico o físico) para transmitir información entre procesos.
    - También el lenguaje debe proveer un protocolo adecuado.
    - Para que la comunicación sea efectiva los procesos deben "saber" cuándo tienen mensajes para leer y cuando deben trasmitir mensajes.

7-

a) Defina sincronización. Explique los mecanismos de sincronización que conozca.

La sincronización es la posesión de información acerca de otro proceso para coordinar actividades.
Los procesos se sincronizan:

- Por exclusión mutua.
    - Asegura que sólo un proceso tenga acceso a un recurso compartido en un instante de tiempo.
    - Si el programa tiene secciones críticas que pueden compartir más de un proceso, la exclusión mutua evita que dos o más procesos puedan encontrarse en la misma sección crítica al mismo tiempo.
- Por condición.
    - Permite bloquear la ejecución de un proceso hasta que se cumpla una condición dada.

b) ¿En un programa concurrente pueden estar presentes más de un mecanismo de sincronización? En caso afirmativo, ejemplifique

Si, pueden estar presentes mas de un mecanismo de sincronización. Un ejemplo de esto es el problema de un "buffer limitado" compartido entre productores y consumidores. En este escenario, se utilizan mecanismos de exclusión mutua para garantizar que un productor o consumidor acceda al búfer de manera exclusiva cuando lo necesita. A su vez, se utilizan mecanismos de condición para permitir que los productores esperen si el búfer está lleno o que los consumidores esperen si el búfer está vacío, asegurando así que la sincronización se realice correctamente para mantener la integridad de los datos y evitar bloqueos o inanición.

8- ¿Qué significa el problema de "interferencia" en programación concurrente? ¿Cómo puede evitarse?

La interferencia es una situación en la que un proceso toma una acción que invalida las suposiciones hechas por otro proceso.
Para evitar la interferencia en programación concurrente se utilizan las acciones atómicas y técnicas de sincronización. Las operaciones atómicas son aquellas que se ejecutan como una única unidad indivisible, lo que significa que no pueden ser interrumpidas por otros procesos. Esto garantiza que, cuando un proceso está realizando una operación atómica en un recurso compartido, ningún otro proceso puede interferir en medio de esa operación.

9- ¿En qué consiste la propiedad de "A lo sumo una vez" y qué efecto tiene sobre las sentencias de un programa concurrente? De ejemplos de sentencias que cumplan y de sentencias que no cumplan con ASV

Una sentencia de asignación x = e satisface la propiedad de "A lo sumo una vez" si:
1) e contiene a lo sumo una referencia crítica y x no es referenciada por otro proceso, o
2) e no contiene referencias críticas, en cuyo caso x puede ser leída por otro proceso.

Una expresiones e que no está en una sentencia de asignación satisface la propiedad de "A lo sumo una vez" si no contiene más de una referencia crítica.

Básicamente a lo sumo una variable compartida y a lo sumo se referencia una vez.

Si una sentencia de asignación cumple la propiedad ASV, entonces su ejecución parece atómica, pues la variable compartida será leída o escrita sólo una vez.

Ejemplos:

- `int x=0, y=0;`
  `co x=x+1 // y=y+1 oc;`
  No hay ref. críticas en ningún proceso.
  En todas las historias ***x = 1*** e ***y = 1***

- `int x = 0, y = 0;`
  `co x=y+1 // y=y+1 oc;`
  El 1er proceso tiene 1 ref. crítica. El 2do ninguna.
  Siempre ***y = 1*** y ***x = 1 o 2***

- `int x = 0, y = 0;`
  `co x=y+1 // y=x+1 oc;`
  Ninguna asignación satisface ASV.
  Posibles resultados: ***x= 1 e y =2*** / ***x = 2*** e ***y = 1***
  Nunca debería ocurrir x = 1 e y = 1 → ERROR

10- Dado el siguiente programa concurrente:

```
x = 2; y = 4; z = 3;
co
   x = y - z // z = x * 2 // y = y - 1
oc
```

a) ¿Cuáles de las asignaciones dentro de la sentencia co cumplen con ASV?. Justifique claramente.

x = y – z no cumple porque se están usando 3 variables compartidas
z = x * 2 no cumple porque se están usando 2 variables compartidas
y = y -1 cumple porque se esta usando una variable compartida y es referenciada una vez

b) Indique los resultados posibles de la ejecución
Nota 1: las instrucciones NO SON atómicas.

Nota 2: no es necesario que liste TODOS los resultados, pero si los que sean representativos de las diferentes situaciones que pueden darse.

X = Y – 1

1) Load posMemY, acum
2) Sub 1, acum
3) Store acum, posMemX

Z = X * 2

4) Load posMemX, acum
5) Add posMemX, acum
6) Store acum, posMemZ

Y = Y – 1

7) Load posMemY, acum
8) Sub 1, acum
9) Store acum, posMemY

Resultados posibles

1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 → Z = 4, X = 2, Y = 3

1 – 2 – 4 – 5 – 6 – 3 – 7 – 8 – 9 → Z = 4, X = 2, Y = 3

7 – 8 – 9 – 1 – 2 – 3 – 4 – 5 – 6 → Z = 2, X = 1, Y = 3

7 – 8 – 9 – 1 – 2 – 4 – 5 – 3 – 6 → Z = 4, X = 1, Y = 3

4 – 7 – 8 – 9 – 1 – 2 – 3 – 5 – 6 → Z = 3, X= 1, Y = 3

11- Defina acciones atómicas condicionales e incondicionales. Ejemplifique.

Acciones atómicas incondicionales: son acciones que se ejecutan sin considerar ninguna condición previa. Siempre se realizan de manera atómica, independientemente de las circunstancias. < x = 1 >

Acciones atómicas condicionales: son acciones atómicas que se ejecutan cuándo se cumple una cierta condición previa. < await S; x = 1 > (puede ser solo < await S >)

12- Defina propiedad de seguridad y propiedad de vida.

Seguridad (safety)

- Esta propiedad se refiere a la garantía de que "nada malo le ocurre a un proceso" o, en otras palabras, que no se producen estados inconsistentes o resultados incorrectos debido a la ejecución concurrente.
    - Una falla de seguridad indica que algo anda mal.
    - Ejemplos de propiedades de seguridad: exclusión mutua, ausencia de interferencia entre procesos, partial correctness(garantiza que si el programa termina, el resultado es correcto. No garantiza que termine).

Vida (liveness)

- Esta propiedad se enfoca en asegurarse de que los procesos no queden bloqueados (deadlocks) y que las actividades puedan avanzar y completarse.
    - Una falla de vida indica que las cosas dejan de ejecutar.
    - Ejemplos de vida: terminación (asegura que el programa termina, pero no garantiza el resultado correcto), asegurar que un pedido de servicio será atendido, que un mensaje llega a destino, que un proceso eventualmentealcanzará su SC, etc $\Rightarrow$ dependen de las políticas de scheduling.

13- ¿Qué es una política de scheduling? Relacione con fairness. ¿Qué tipos de fairnes conoce?

Una **política de scheduling** se refiere a un conjunto de reglas y algoritmos utilizados para determinar cuál será el próximo proceso o hilo que se ejecutará en un sistema concurrente o en un sistema operativo. Cuando se habla de ejecutar el próximo proceso o hilo, se esta hablando de ejecutar las acciones atómicas de dicho proceso.

Fairness se relaciona con la equidad en la ejecución de procesos o hilos en un sistema concurrente. Se esfuerza por garantizar que todos los procesos tengan la oportunidad de avanzar y realizar sus operaciones, evitando situaciones de bloqueo o inanición.

Fairness trata de garantizar que los procesos tengan chance de avanzar, sin importar lo que hagan los demás, es decir, que se ejecuten acciones atómicas de todos lo procesos.

Tipos de Fairness:

Fairness incondicional: significa que todas las tareas que están listas para ejecutarse eventualmente se ejecutan, sin importar si son condicionales o no.

Fairness débil: significa que además de la fairness incondicional, las tareas condicionales que se vuelven elegibles se ejecutan, siempre y cuando su condición se mantenga verdadera hasta que la tarea sea asignada.

Fairness fuerte: significa que además de la fairness incondicional, las tareas condicionales que se vuelven elegibles se ejecutan, incluso si su condición cambia de valor muchas veces.

14- ¿Por qué las propiedades de vida dependen de la política de scheduling? ¿Cómo aplicaría el concepto de fairness al acceso a una base de datos compartida por n procesos concurrentes?

La propiedad de vida puede verse afectada, si por ejemplo: utilizacemos una politica de prioridades(alta,media,baja) en el acceso de la BD, si constantemente te llegan procesos de prioridad alta, los de baja prioridad pueden parecer inanicion. Utilizar scheduling como RoundRobin puede mejorar la asignacion equitativa de tiempo de procesador para cara proceso. Al igual que utilizar politicas de AGING donde cada determinado tiempo de procesador, se reevaluen las prioridades de procesos pudiendo incrementar su prioridad.

15- Dado el siguiente programa concurrente, indique cuál es la respuesta correcta (justifique claramente)
int a = 1, b = 0;
co await (b = 1) a = 0 // while (a = 1) { b = 1; b = 0; } oc
a) Siempre termina
b) Nunca termina
c) Puede terminar o no

Respuesta de la Banda: C- El proceso puede terminar, para que ello ocurra se debe ejecutar el proceso 2 while (a = 1) { b = 1; b = 0; }; y cambie de proceso cuando el valor de B=1; en ese caso el proceso 1 de forma atómica modificaria el valor de A=0; asegurando que cuando se cambie al proceso 2 que por la evaluacion del while, corte su ejecución y el programa finalice.
La politica de fainess Fuerte nos asegura que va a terminar en algun momento, mientras que la DEBIL puede que si o no.