Cuestionario teoría 1 a 2

sábado, 19 de septiembre de 2020 11:20

1) Mencione al menos 3 ejemplos donde pueda encontrarse concurrencia

El cuerpo humano, un auto, una canción.

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

Concurrencia: es una forma de trabajar que intercala las tareas y, si es posible, las divide en los diferentes actores que pueden ejecutarlas.
Definición de la cátedra: concurrencia es la capacidad de ejecutar múltiples actividades en paralelo o simultáneamente.

Procesamiento secuencial: es una manera de ejecutar procesos de a 1 por vez de manera ordenada, sin poder intercalarlos.
Definición de la cátedra: un único flujo de control que ejecuta una instrucción y cuando esta finaliza ejecuta la siguiente.

Procesamiento concurrente: es la posibilidad de ejecutar de manera concurrente un proceso, intercalandolos en un mismo procesador e incluso dividiendo las tareas en varios procesadores.
Definición de la cátedra: un programa concurrente especifica dos o más "programas secuenciales" que pueden ejecutarse concurrentemente en el tiempo como tareas o procesos.

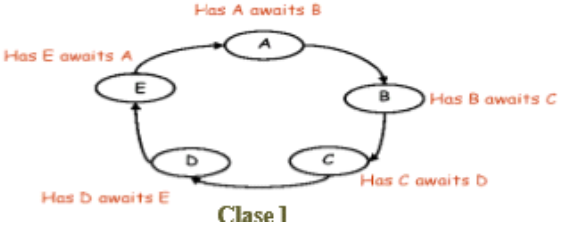
Procesamiento paralelo: procesamiento concurrente pero de manera simultánea en varios procesadores.
Definición de la cátedra: se asocia con la ejecución concurrente en múltiples procesadores con el objetivo principal de reducir el tiempo de ejecución.

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

Dos (o más) procesos pueden entrar en deadlock, si por error de programación ambos se quedan esperando que el otro libere un recurso compartido.

4 propiedades necesarias y suficientes para que exista deadlock son:

- Recursos reusables serialmente: los procesos comparten recursos que pueden usar con exclusión mutua.
- Adquisición incremental: los procesos mantienen los recursos que poseen mientras esperar adquirir recursos adicionales.
- No-preemption: una vez que son adquiridos por un proceso, los recursos no pueden quitarse de manera forzada sino que sólo son liberados voluntariamente.
- 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.



Clase 1 34

4) Defina inanición. Ejemplifique.

La inanición ocurre cuando un proceso no puede adquirir un recurso compartido y, por lo tanto, no puede continuar su ejecución. Esto puede pasar, por ejemplo, teniendo un algoritmo en donde se decida a quien darle el recurso en base a una prioridad fija, el proceso que tenga baja prioridad, al ser fija, puede que nunca llegue a adquirir el recurso.

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

Entiendo por no determinismo la imposibilidad de determinar un resultado de antemano. En la ejecución concurrente ocurre seguido cuando tenemos varios procesos para ejecutar y sin un orden de prioridades o un flujo restrictivo, la traza de ese programa puede variar con cada ejecución de manera no determinista, ya que no podemos determinar cuál instrucción se tomará primero y cuál luego.
Definición de la cátedra: no hay un orden preestablecido para la ejecución (ejecuciones con la misma entrada pueden

generar diferentes salidas).

6) Defina comunicación. Explique los mecanismos de comunicación que conozca
La comunicación es la forma de intercambiar información entre procesos.
Hay 2 mecanismos:

Memoria compartida: cuando los procesos que se quieren comunicar utilizan la misma memoria, pueden escribir y leer celdas de la misma para comunicarse.
A través de red: cuando los procesadores se encuentran en diferentes PC o simplemente no comparten memoria, pueden generar un canal de comunicación que va a funcionar como red para conectarse los unos con los otros.

Definición de la cátedra:

La comunicación entre procesos concurrentes indica el modo en que se organizan y trasmiten datos entre tareas concurrentes.

- **Memoria compartida**
  - Los procesos intercambian información sobre la memoria compartida o actúan coordinadamente sobre datos residentes en ella.
- **Pasaje de mensajes**
  - Es necesario establecer un canal (lógico o físico) para transmitir información entre procesos.
  - También el lenguaje debe proveer un protocolo adecuado.

7- a) Defina sincronización. Explique los mecanismos de sincronización que conozca.
7) b) ¿En un programa concurrente pueden estar presentes más de un mecanismo de sincronización? En caso afirmativo, ejemplifique

a)
Sincronización: la sincronización es la condonación de actividades entre procesos, usando información especial para lograrlo.
Hay 2 mecanismos de sincronización:

Sincronización por exclusión mutua: cuando un proceso toma un recurso nadie más puede usarlo.
Sincronización por condición: un proceso no puede acceder a un recurso hasta que se cumpla una condición dada.

b) Si que puede pasar, por ejemplo, se puede hacer una sincronización por condición para filtrar un número menor de procesos con, por ejemplo, mayor prioridad, y entre esos usar sincronización por exclusión para acceder al recurso.

8) ¿Qué significa el problema de "interferencia" en programación concurrente? ¿Cómo puede evitarse?
Interferencia: un proceso toma una acción que invalida las suposiciones hechas por otro proceso.
Se puede evitar haciendo atómicas las acciones críticas (atomicidad de grano grueso).

9)

Referencia crítica en una expresión ⇒ referencia a una variable que es modificada por otro proceso.

Asumamos que toda referencia crítica es a una variable simple leída y escrita atómicamente.

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.

En resumen, una sentencia cumple la condición de ASV cuando hay a lo sumo una variable compartida y está referenciada a lo sumo 1 vez.
Esto nos asegura que la sentencia se ejecute como si fuese atómica.

- `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.
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.

a) x= y - z y z = x*2 son las asignaciones que no cumple con la propiedad ASV, ya que hay 2 variables compartidas en lugar de 1 (x, z).

b)

X	Z	Y
1	2	3
-1	4	3
0	0	3
0	4	3
1	4	3
0	4	3

11) Defina acciones atómicas condicionales e incondicionales. Ejemplifique.

Las acciones atómicas se pueden definir con la sentencia de sincronización await. La misma puede tener condiciones y sentencias. Si solo tiene condición (acción atómica condicional), es una exclusión por condición, ya que si un proceso no cumple la condición no sigue su ejecución. Si solo tiene sentencias, es una exclusión mútua (acción atómica incondicional).
Ejemplos:

- ***Await general:*** ⟨**await (s>0) s=s-1;**⟩
- ***Await para exclusión mutua:*** ⟨**x = x + 1; y = y + 1**⟩
- ***Ejemplo await para sincronización por condición:*** ⟨**await (count > 0)**⟩

  Si B satisface ASV, puede implementarse como ***busy waiting*** o ***spin loop***
  do (not B) → skip od **(while (not B);)**

12) Defina propiedad de seguridad y propiedad de vida.

La propiedad de seguridad asegura estados consistentes, que no van a ocurrir fallas entre los procesos.
La propiedad de vida asegura que el programa va a terminar, que no van a ocurrir deadlocks ni nada que detenga la ejecución.

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

La política de scheduling determina cuál será la próxima acción atómica, de las elegibles, en ejecutarse.
Si una política de scheduling es mala, puede que un proceso sufra de inanición.

La política de scheduling se pueden dividir según las condiciones de fairness que asegura:

Fairness Incondicional. Una política de scheduling es incondicionalmente fair si toda acción atómica incondicional que es elegible eventualmente es ejecutada.

Fairness Débil. Una política de scheduling es débilmente fair si :

(1) Es incondicionalmente fair y

(2) Toda acción atómica condicional que se vuelve elegible eventualmente es ejecutada, asumiendo que su condición se vuelve ***true*** y permanece ***true*** hasta que es vista por el proceso que ejecuta la acción atómica condicional.

Esto no asegura que cualquier sentencia await con condición eventualmente se ejecutará, puede ocurrir que mientras el recurso esté ocupado, el valor de la condición pase a true pero vuelva a false.

Fairness Fuerte. Una política de scheduling es ***fuertemente fair*** si:

(1) Es incondicionalmente fair y

(2) Toda acción atómica condicional que se vuelve elegible eventualmente es ejecutada pues su guarda se convierte en ***true*** con infinita frecuencia.