###### Licenciatura en Ingeniería de Software

###### Programación Visual III

|  |  |
| --- | --- |
| UNIDAD II | Tarea # PVT8 |
| Nombre: Miguel Alejandro Guevara Mendoza | Fecha: 09/11/2018 |

Glosario # 2

**Concurrencia**

Se conoce como concurrencia a la rama de la informática que trata de las técnicas de programación que se usan para expresar el paralelismo entre tareas y para resolver los problemas de comunicación y sincronización entre procesos.

**Procesamiento en Paralelo**

El procesamiento en paralelo es la ejecución simultánea de la misma tarea en dos o más microprocesadores con el fin de obtener resultados más rápidos. Los recursos de la ordenadora pueden incluir un solo equipo con varios procesadores, o un número de computadoras conectadas por una red, o una combinación de ambos, y los procesadores acceden a los datos a través de una memoria compartida.

**future**

Un futuro es un objeto que permite que un hilo pueda devolver un valor a la sección de código que lo invocó.

**Race Condition**

La condición de carrera (race condition) ocurre cuando dos o más procesos acceden un recurso compartido sin control, de manera que el resultado combinado de este acceso depende del orden de llegada. Suponga, por ejemplo, que dos clientes de un banco realizan cada uno una operación en cajeros diferentes al mismo tiempo.

**Deadlock**

Es el bloqueo permanente de un conjunto de procesos que compiten por los recursos del sistema o bien se comunican unos con otros. A diferencia de otros problemas de la gestión concurrente de procesos, para el caso general no existe una solución eficiente. En esta sección, se examinará la naturaleza del problema del interbloqueo

**Exclusion Mutua**

Exclusión Mutua es la comunicación requerida entre dos o más procesos que se están ejecutando en paralelo y que necesitan a la vez el uso de un recurso no compartible. Consiste en asignar el recurso no compartible a sólo uno de los procesos, mientras que los otros deben permanecer a la espera hasta que finalice la utilización de dicho recurso por el proceso al que se le asigno. Cuando este proceso termine, el recurso será asignado a uno de los procesos en espera.

**Seccion Critica**

Una región o sección crítica es una secuencia de instrucciones que no debe ser interrumpida por otros procesos, es decir, se debe tratar una región crítica como una sola instrucción atómica.

**Sincronización**

Se usa para indicar que ciertas partes del código, (habitualmente, una función miembro) están sincronizadas, es decir, que solamente un subproceso puede acceder a dicho método a la vez.  
  
Cada método sincronizado posee una especie de llave que puede cerrar o abrir la puerta de acceso. Cuando un subproceso intenta acceder al método sincronizado mirará a ver si la llave está echada, en cuyo caso no podrá accederlo. Si método no tiene puesta la llave entonces el subproceso puede acceder a dicho código sincronizado.

**Sleep, Wait, Task.Wait**

sleep: Permite hacer una pausa por una determinada cantidad de segundos en la ejecución de un programa.

wait: deja bloqueado al hilo que lo llama

Task.wait: Puede evitar cuellos de botella de rendimiento y mejorar la capacidad de respuesta total de la aplicación mediante la programación asincrónica

**Lock, Mutex y SpinLock**

lock: instrucción lock de C# para sincronizar el acceso de un subproceso al recurso compartido.

mutex: se usan en programación concurrente para evitar que entre más de un proceso a la vez en la sección crítica.

spinlock: Los spinlocks se aseguran de que solo un único thread entra en una sección crítica a la vez. Cualquier otro thread que quiera entrar en la sección crítica espera hasta que el thread que está dentro salga.

**Semaphore, SemaphoreSlim y los read/write locks.**

Semaphore: Un semáforo es una forma de limitar las tareas que pueden operar en forma simultánea en un recurso compartido (protegido). Los recursos protegidos o partes de código críticos pueden incluir las variables globales o comunicaciones con instrumentos externos.Los semáforos son muy similares a los "Mutex" (Objeto de exclusión mutua ) en otros lenguajes de programación.

SemaphoreSlim: La clase SemaphoreSlim representa un semáforo ligero y rápido que se puede usar para esperar en un único proceso cuando se supone que los tiempos de espera serán muy cortos. SemaphoreSlim se basa en la medida de lo posible en las primitivas de sincronización proporcionadas por Common Language Runtime (CLR).

Read/write locks: La interfaz Lock esta concebida para evitar la exclusión mutua que previene la actualización concurrente del bloque que protege. Garantiza la exclusión mutua sin diferenciar las  
situaciones writer/writer y writer/reader que son incompatibles y las reader/reader que si son compatibles La interfaz ReadWriteLock diferencia dos tipos de lock uno  
para writer y otro reader.

**Polling**

El método de polling (llamar a una estación para que transmita o se disponga a recibir un mensaje), requiere un control centralizado de todas las estaciones de la red.

**Blocking**

El hilo del proceso se bloquea y el procesador queda libre.

**Spinning**

El hilo del proceso se bloquea pero no libera al procesador. Queda a la espera hasta que se desbloquee.

**Latency**

La latency es el tiempo que transcurre entre que se empieza a preparar un paquete, se envía por la red y es recibido.

**Overhead**

Es la cantidad de tiempo requerido para coordinar tareas paralelas, en lugar de hacer un trabajo útil. Incluye factores como: Tiempo de inicio de la tarea o Sincronización

**Context Switch**

Es la acción que efectúa el scheduler cuando transfiere el procesador de un proceso a otro. Para realizar el cambio de contexto, el scheduler debe realizar diversas labores

**Locking**

La instrucción lock adquiere el bloqueo de exclusión mutua de un objeto determinado, ejecuta un bloque de instrucciones y luego libera el bloqueo. Mientras se mantiene un bloqueo, el subproceso que lo mantiene puede volver a adquirir y liberar el bloqueo. Ningún otro subproceso puede adquirir el bloqueo y espera hasta que se libera.

**Mutex**

Se usan en programación concurrente para evitar que entre más de un proceso a la vez en la sección crítica.