###### Licenciatura en Ingeniería de Software

###### Programación Visual III

|  |  |
| --- | --- |
| 2do Parcial | Glosario |
| Nombre: Kevin Ricardo Ceja Ramos | Fecha: 28/11/2018 |

**Concurrencia**

Dos o más procesos decimos que son concurrentes, paralelos, o que se ejecutan concurrentemente, cuando son procesados al mismo tiempo, es decir, que para ejecutar uno de ellos, no hace falta que se haya ejecutado otro.

**Procesamiento en Paralelo**

Aquellos procesos que se ejecutan y/o procesan a la vez, anteponiéndose a los procesos lineales o secuenciales, que serán ejecutados de uno en uno.

**Future**

Un objeto o un futuro esperado que ocurrirá en algún momento.

**Race Condition**

Un evento inesperado que ocurre cuando múltiples entidades intentan acceder al mismo dato compartido, El sistema trabaja como se espera, pero algunas veces por los delays, estas operaciones pueden arrojar resultados inesperados.

**DeadLock**

Una situación donde el acuerdo de un argumento no puede ser alcanzado por ninguna de las 2 partes, y ninguna de las 2 partes se rinde o cede el paso, esto ocasiona un ciclo sin fin.

**Exclusion Mutua**

La exclusión mutua es la actividad que realiza el sistema operativo para evitar que dos o más procesos ingresen al mismo tiempo a un área de datos compartidos o accedan a un mismo recurso.

**Seccion Critica**

Se refiere a donde está concentrado las instrucciones importantes de nuestro programa, donde esta basado la mayor parte de nuestro código, esta sección tiene que funcionar sin importar que.

**Sincronización**

La sincronización entre procesos puede definirse como la necesidad que tienen algunos procesos de bloquearse en determinadas circunstancias y ser despertados cuando ocurren ciertos eventos.

**Sleep, Wait, Task.Wait**

Sleep: Hace una pausa por determinados MS, duerme al thread hasta que los MS pasen.

Wait: Deja bloqueado al thread.

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 Thread se bloquea y el procesador queda libre.

**Spinning**

Deja al Thread en espera, a diferencia del Blocking el procesador se queda en espera.

**Latency**

El tiempo de respuesta que hay entre una conexión, el tiempo de llegada y respuesta.

**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**

Instrucción de c# que es lock, adquiere el bloqueo de exclusión mutua de un objeto determinado, y ejecuta una serie de instrucciones y después libera el bloqueo. Ningún otro subproceso puede adquirir el bloqueo hasta que se libere.

**Mutex**

De la programación concurrente para evitar un bloqueo en la sección critica.