**Programacion Concurrente**

**CLASE 1**

**Ejecucion de Codigo:**

**No determinístico:** no hay un orden de ejecucion de código.

**Secuencial:** ejecuta un codigo y hasta no terminar no cambia.

**Concurrencia:** se puede ir trabajando sobre distintos procesos alternando entre ellos.

**Concurrente con paralelismo:** Se puede ir ejecutando mas de un proceso al mismo tiempo y también cada Core puede ir cambiando de proceso. (El paralelismo depende del hardware).

**Procesos livianos, threads o hilos:**

Son parte de un proceso, todos los hilos de un proceso comparten espacio de dirección.

Entre ellos se pueden comunicar y acceder a las mismas variantes.

Hacer un context switch entre hilos es más fácil / rápido que hacerlo entre procesos.

Los procesos no comparten espacio de memoria

Las librerias comparten hilos

**Comunicacion entre procesos**

**Por memoria compartida**

Se debe tener una memoria compartida psra pasarse los datos, tiene mas chance a error por que no se sabe si ya se escribio el dato ose sobre escribio o lo que sea. No funciona en cluster de computadoras

**Por pasaje de mensajes**

Por comunicacion entre procesos

**Tipos de comunicacion**

**Por condición:**

Se permite la ejecucion de un proceso hasta que se cumpla una condición. Ejemplo tipico un buffer donde hay un productor y consumidor, el productor pone elementos y el consumidor saca. Si el buffer esta lleno solo puede competir los consumidores, si hay espacio cualquiera de los 2. Pero siempre uno solo por vez

**Por exlusion mutua:**

Hay un recurso compartido que puede ser usado por un proceso por vez. Se debe elegir un proceso para que lo utilice. Se llama seccion critica al recurso compartido que debe ser usado por un solo proceso

**Interferencia:**

Es un error. Cuando un proceso realiza una accion en memoria compartida que invalida las suposiciones hechas por otro procesos. Ejemplo con una condicion chequeas que sea dstinto de 0 y mientras se ejecuta dentro de la condicion otro proceso/hilo modifica el valor a 0

Otro ejemplo prodria se cuando se incrementa una variable global por dos procesos al mismo tiempo, se puede dar el problema que ambos copien la variable a memoria para incrementar y ambos incrementen de 5 a 6 en vez de 5 a 6 y 6 a 7

**Granuladidad:**

-Define en cuantos cachitos de va a dividir el proceso.

-Grano fino es que el procesos se va a dividir en muchos cachos.

-Grano grueso en menos cantidad de procesos.

-Muchos cachos va a requerir de mayor uso comunicacion o sincronización.

Un grano grueso seria ideal para sistemas de poco procesadores pero mas potentes, con una comunicación entre maquina lenta.

De grano fino ideal con muchos procesadores con mayor velocidad de comunicacion pero no tan potentes.

**Administracion de recursos compartidos**

Hay que asegurar que no haya deadlock, no haya inanicion, ni dos al mismo tiempo (exclusion mutua) y fairness.

Fairness: Es la capacidade de distribuir equitativamente el uso del recurso compartido

**Deaadlock:**

Para que exista deadlock debe:

**-Recursos reutilizables serialmente:**Los procesos comparten recursos que pueden usar con exlusion permanente

**-Adquisición incremental:**Se agsrra el conjunto de recirsos necesarios al mismo tiempo, o se agarra todo o se agarra nada

**-No-preemption:**Si hay un recirso de los requeridos libres se sgarra ese recirso hasta esperar que los otros necesarios se liberen tambien. Pasado un tiemp ode espera se liberan el tecirso que esta bloqueando

**-Espera cíclica:**Existe una cadena circular donde cada proceso está esperando el recurso siguient epara liberar el actual. Se soluciona empezando todos por el mismo recurso.

**Lenguajes de programación concurrente**

Deben permitir definir procesos que forman partes del programa y herramientas que permitan la comunicacion, implementacion y sincronizacion de procesos

Problemas de la programacion concurrente:

-Compartir recursos (principalmente)

-No determinismo

-Overheard de context switch (no tanto por que siempre existe), comunicacion o sincronización

-Mayor tiempo de desarrollo y puesta a punto

-Adaptacion del software paralelo al hardware en uso

**Paralelismo:**

El paralelismo es una parte de la concurrencia

Siendo concurrencia cuando partimos un programa en partes y ejecutamos en partes

El paralelismo es la ejecucion de la concurrencia en minimo dos procesadores.

**Instrucciones usuales de paralelismo:**

**-Skip:** se usa para completar una estructura, termina directamente.  
Se selecciona todos los if verdaderos y se ejecuta uno de manera no deterministica

If B1 -> A1

[] B2 -> A2

fi

**-Skip variante repeat-until**   
Funciona igual que el if, se ejecuta hasta que todas las condiciones sean falsas

Do B1 -> A1

[] B2 -> A2

Od

**-Concurrencia:**   
Indica que se va a necesitar ejecutar de forma concurrente S1, S2, Sn. Se termina cuando todas las sentencias terminaron si ejecucion

Co S1 // S2 // Sn oc

Cuantificador para hacer N veces un mismo proceso:

co [i=1 to n| { ali]=0; b[i]=0 } oc

Otra forma de marcar concurrencia

Process A {sentencia}

Cuantificador:

Process B [i=0 to N] {sentencia}

**-Diferencia:**