Según la Real Academia Española, concurrencia es el concurso de varios sucesos en un mismo tiempo.

Concurrencia es la capacidad de ejecutar múltiples actividades en paralelo o simultáneamente

Permite a distintos objetos actuar al mismo tiempo

Dónde encontramos concurrencia?

- Los sistemas biológicos suelen ser masivamente concurrentes: comprenden un gran número de células, evolucionando simultáneamente y realizando (independientemente) sus procesos. En el mundo biológico los sistemas secuenciales rara vez se encuentran.
- En algunos casos se tiende a pensar en sistemas secuenciales en lugar de concurrentes para simplificar el proceso de diseño, pero la necesidad de sistemas de cómputo cada vez más poderosos y flexibles atenta contra la simplificación de asunciones de secuencialidad
- "Concurrency is everywhere"
- Todas las entidades, sean plantas, animales, máquinas, etc, ejecutan en paralelo unas con otras, cualquier sistema más o menos "inteligente"exhibe concurrencia…
- Navegador Web accediendo una página mientras atiende al user, Varios navegadores accediendo a la misma página
- Acceso de varias aplicaciones a disco para guardar
- El teléfono avisa recepción de llamada mientras se habla
- Varios usuarios conectados al mismo sistema (x ej, haciendo una reserva)

Otros??

Un poco de historia de la concurrencia

Evolución en respuesta a los cambios tecnológicos...

De enfoques ad-hoc iniciales a técnicas generales de programación

-60's.Evolución de los SO.

Controladores de dispositivos (canales) operando independientemente de un procesador permitiendo E/S .

Interrupciones. No determinismo.

Multiprogramación. Problema de la sección crítica.

-70's.Formalización de la concurrencia en los lenguajes

-80's.Redes, procesamiento distribuido

-90's.MPP, Internet, C/S, Web computing...

-Hoy .procesamiento masivo de datos distribuidos, SDTR, computación móvil, Cluster y multicluster computing, sistemas colaborativos, computación ubicua, computación pervasiva, grid computing, cloud computing…

1. Conceptos de programación concurrente

Un programa es un conjunto de instrucciones, que indica que hacer con un conjunto de datos de entrada para producir algún tipo de salida. Es estático. En programación orientada a objetos puede compararse con el concepto de clase. Para que un programa pueda hacer algo de verdad debemos ponerlo en ejecución.

Un proceso es una entidad dinámica, pude compararse con el concepto de objetos en el ámbito de la POO. Está representado por el valor del contador del programa, contenido de los registros de del procesador, una pila, y una sección de datos que contiene variables globales.

Múltiples procesos pueden corresponder a un programa, ejemplo: un servidor de aplicaciones donde reside una aplicación de navegador de internet y existen varios usuarios ejecutando ese navegador en sitios web diferentes. en la figura 1.1, puede observarse un programa almacenado en disco y 3 instancias de ese programa ejecutándose. Son 3 procesos, cada uno con su propia información.

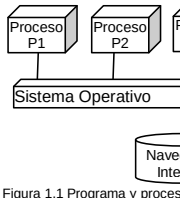


Figura 1.1 Programa y proces

Si sustituimos suceso por proceso tenemos una primera aproximación a la definición de programación concurrente.

Dos procesos son concurrentes cuando existe un solapamiento en la ejecución de sus instrucciones. No tiene que ejecutarse exactamente al mismo tiempo, es suficiente que exista un intercalado de ejecución de instrucciones, la programación concurrente es un paralelismo potencial.

El caso de la figura anterior, se trata de un primer nivel de concurrencia, existen 3 procesos independientes que se ejecutan al mismo tiempo sobre un sistema operativo, cada proceso corresponde a una instancia de un programa. El programa de navegador de internet puede dar lugar a más de un proceso, uno que controle las acciones del usuario con la interfaz, otro que hace las peticiones al servidor, etc. De esta forma la figura 1.1 se convertiría en la figura 1.2 suponiendo que se crean 2 procesos cada vez que se ejecuta el programa, y todos estos procesos pueden ejecutarse concurrentemente.

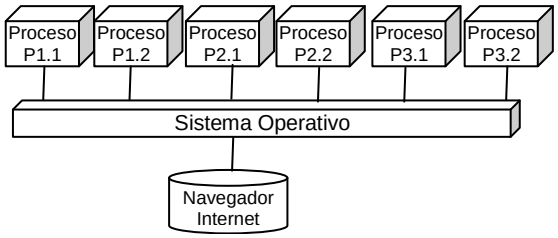


Figura 1.2. Un programa dando lugar a más de un proceso

Cuando varios procesos se ejecutan concurrentemente, puede haber procesos que colaboren para un determinado fin, otros que compitan recursos del sistema, incluso aquellos que colaboran deberán competir a la hora de obtener tiempo de procesador.

Para realizar las tareas de colaboración y competencia por los recursos es necesario introducirnos a los mecanismos de comunicación y sincronización entre procesos. Precisamente del estudio de estos mecanismos se trata la programación concurrente.

En resumen "la Programación concurrente: es la disciplina que se encarga del estudio de las notaciones que permiten especificar la ejecución concurrente de las acciones de un programa, las técnicas para resolver los problemas inherentes a la ejecución concurrente, que son la comunicación y sincronización"

Objetivos de los sistemas concurrentes:

Ajustar el modelo de arquitectura de hardware y software al problema del mundo real a resolver. El mundo real ES CONCURRENTE

Incrementar la performance, mejorando los tiempos de respuesta de los sistemas de procesamiento de datos, a través de un enfoque diferente de la arquitectura física y lógica de las soluciones.

1.2 Beneficios o ventajas de la programación concurrente:

Existen diferentes motivos por los que la programación concurrente es útil. Destacándose la Velocidad de ejecución que se puede alcanzar, mejor utilización de la CPU de cada procesador, y Solución a problemas de naturaleza concurrente. (la explotación de la concurrencia inherente a la mayoría de los problemas reales).

Velocidad de ejecución: al asignar un proceso a cada procesador, el programa se ejecuta más rápido. Los programas de cálculo son los más beneficiados de este hecho.

Solución de problemas inherentemente concurrente:

a) Sistemas de control:

Son aquellos en los que hay una captura de datos, generalmente con sensores, un análisis de esos datos y una posible actuación posterior en función del resultado del análisis. La recolección puede hacerse en diferentes entidades físicas. Tanto la captura de datos desde cada entidad, su tratamiento y actuación son candidatos a ser procesos distintos de naturaleza concurrente. Lo que garantiza que el sistema de control pueda responder a alarmas que se produzcan.

b) Tecnologías web:

La mayoría de los programas relacionados con la web son concurrentes. Los servidores web son capaces de atender concurrentemente múltiples conexiones de usuarios, programas de chat (que permitan conversación de varios usuarios); los servidores de correo (permiten que múltiples usuarios puedan enviar y recibir mensajes al mismo tiempo), navegadores que permiten que usuarios puedan hacer un descargar mientras navegan, ejecutan un applet de java, etc.

c) Aplicaciones basadas en interfaces de usuarios:

La concurrencia va a permitir que el usuario pueda interactuar con la aplicación aunque esté realizando alguna tarea que consume mucho tiempo de procesador. un proceso controla la interfaz, mientras que otro hace las tareas que requieren un uso intensivo de CPU. Facilitando que tareas largas puedan se abortadas con facilidad

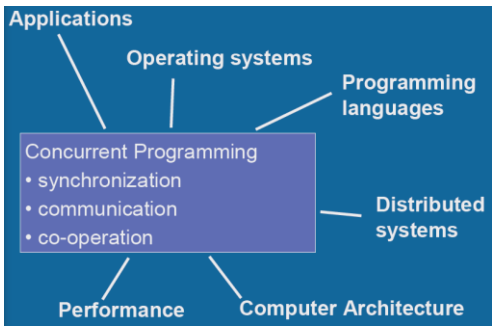
d) Simulación:

La concurrencia va a permitir modelar objetos físicos que tienen un comportamiento autónomo independiente y ponerlos en ejecución de forma independiente y concurrente sincronizados apropiadamente.

e) Sistema de Gestión de Base de Datos (SGBD):

En estos sistemas la concurrencia va a ser muy importante cuando se va a permitir a varios usuarios interactuar con el sistema. . Cada usuario puede ser visto como un proceso. Se debe evitar que varios usuarios modifiquen el mismo registro en el mismo tiempo, y permitir el acceso concurrente para que varios usuarios puedan consultar un mismo instante un registro sin modificarlo.

Conexiones



Por qué es necesaria la Programación Concurrente?

•Ganancia de HW por el uso de multiprocesadores –paralelismo

•Incremento en el rendimiento de la aplicación por mejor uso del HW

•Mejora en la respuesta de la aplicación. Tareas de alta prioridad para los pedidos de usuario. - Estructura más apropiada para programas que controlan múltiples actividades y manejan múltiples eventos

- STR (sistemas de tiempo real) que controlan hardware con múltiples sensores/actuadores. Reacción a entradas asincrónicas

- Base p/ construir sistemas (distribuidos, C/S)

- Multi-core/Multi-procesadores-Tareas "caras"computacionalmente realizadas con HW multi-X

2.1 Concurrencia y arquitecturas hardware

Cuando hablamos del hardware en el que se van a ejecutar los procesos concurrentes, nos referimos a el número de procesadores en el sistema. Se puede entonces hacer una primera distinción: Sistemas Monoprocesador y Sistemas Multiprocesador, en ambos es posible tener concurrencia.

2.1.1 Sistemas Monoprocesador: evidentemente no todos los procesos pueden ejecutarse al mismo tiempo en el procesador, pero el sistema operativo va alternando el tiempo de procesador entre los procesos, dando a los usuarios sensación de ejecutarse simultáneamente. De esta forma, cuando un proceso que está ocupando el procesador necesita hacer una operación de entrada salida, puede abandonar el procesador para que otro pueda

Existen otros posibles beneficios del uso de concurrencia en sistemas monoprocesador:

- La posibilidad de proporcionar un servicio interactivo a múltiples usuarios. Ejem: sistema operativo multiusuario ejecutándose sobre una maquina monoprocesador.
- La posibilidad de dar solución adecuada a problemas que son de naturaleza concurrente.

En un sistema monoprocesador los procesos comparten la misma memoria. Los procesos se sincronizan y comunican mediante variables compartidas.

Como se ve en la figura, cuando un proceso A quiere comunicarle algo al proceso B, lo deja en una variable compartida y conocida por ambos para q B lo pueda leer

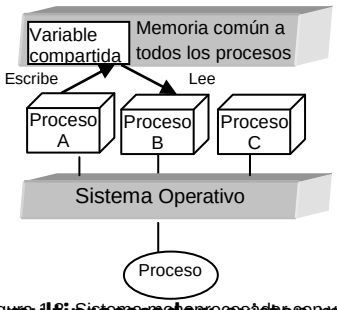


Figura 1.3. Sistema con 1 procesador con variable

2.1.2 **Sistemas multiprocesador.** Existen más de un procesador, permite paralelismo real entre procesos. Como en un sistema concurrente hay más procesos que procesadores, es necesario algún esquema de multiprogramación en uno más procesadores. Los sistemas multiprocesador se clasifican en:

- Sistemas fuertemente acoplados: los procesadores y dispositivos (incluida la memoria) están conectados a un bus. Permitiendo que todos los procesadores puedan compartir la misma memoria. Puede que cada procesador tenga su propia memoria local, pero la sincronización y comunicación se realiza mediante variables situadas en memoria compartida (figura 1.4)

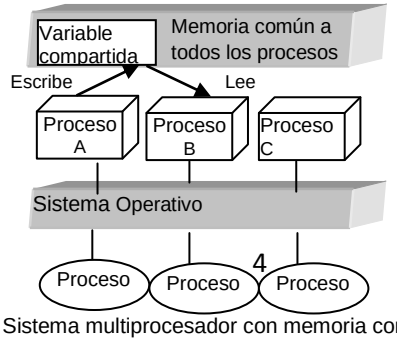


Figura 1.4. Sistema multiprocesador con memoria compartida

- Sistemas débilmente acoplados: cada procesador tiene su memoria local y está conectado con los otros procesadores mediante algún enlace de comunicación. Ejem: sistemas distribuidos. Los nodos están distribuidos geográficamente, conectados de alguna forma. Cada nodo puede ser mono o multiprocesador (internet).
- Multiproceso es la gestión de varios procesos dentro de un sistema multiprocesador, donde cada procesador puede acceder a una memoria común, y procesamiento distribuido es la gestión de varios procesos en procesadores separados cada uno con su memoria local.

Figura 1.5

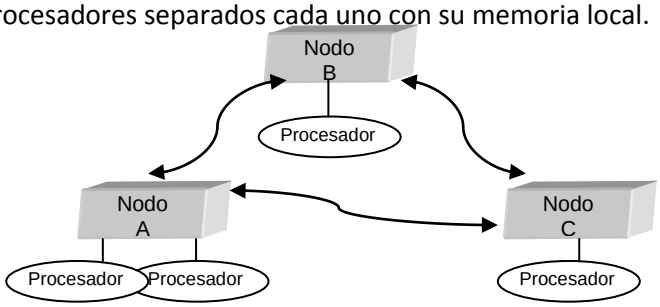


Figura 1.5. Sistema Distribuido

Un programa concurrente define un conjunto de acciones que pueden ser ejecutadas simultá-neamente.

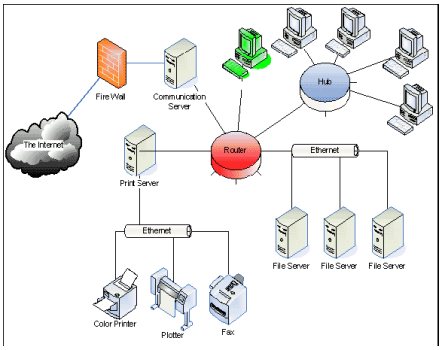
Un programa paralelo es un tipo de programa concurrente diseñado para ejecutarse en un sistema distribuido.

Un programa distribuido: es un programa paralelo que está diseñado, es decir en una red de procesadores autónomos que no comparten memoria en común.

Un programa concurrente debe funcionar independientemente si se ejecuta en un sistema monoprocesador o multiprocesador, ya sea fuerte o débilmente acoplado.

En resumen:

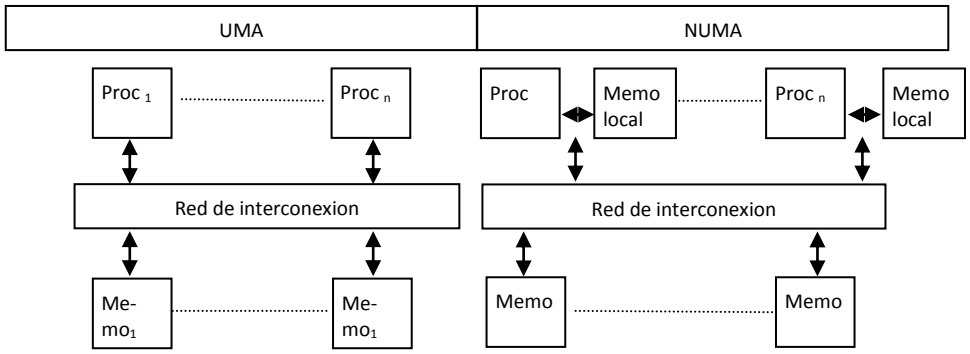
- Procesador
  - Ejecutar varias instrucciones en paralelo
  - Ejecutar varios threads en paralelo
  - Ejecutar varios procesos en paralelo
- Sistema
  - Varios procesadores o procesadores de video (consolas)
  - Varios dispositivos de E/S
- LAN o WAN
  - Varios sistemas (en clusters)
- Internet y otras redes
  - Varios sub-sistemas



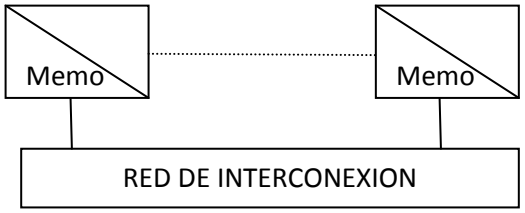
- Límite físico en la velocidad de los procesadores
  - Máquinas “monoprocesador”?? Ya no…
  - Más procesadores por chip para tener mayor potencia de cómputo
  - CPUs Multicore
  - Cómo usarlos eficientemente?.
- Programación concurrente y paralela

- Niveles de memoria y cache. Consistencia.

- Multiprocesadores de memoria compartida.
  La interacción se da modificando datos almacenados en la MC
  Esquemas UMA con bus o crossbar switch(SMP, multiprocesadores simétricos)
  Esquemas NUMA para mayor número de procesadores distribuidos.
  Problema de consistencia



- Multiprocesadores con memoria distribuida.
  Procesadores conectados por una red. C/u tiene memoria local y la interacción es sólo por pasaje de mensajes.
  Grado de acoplamiento de los procesadores:
  -Multicomputadores (tightly coupled machine). Procesadores y red físicamente cerca. Pocas aplicaciones a la vez, cada una usando un conjunto de procesadores. Alto ancho de banda y velocidad.
  -Redes (loosely coupled multiprocessor).
  -NOWs/ Clusters.
  -Memoria compartida distribuida.



Procesamiento secuencial, concurrente y paralelo

Analicemos la solución secuencial y monoprocesador (UNA máquina) para fabricar un objeto compuesto por N partes o módulos.

La solución secuencial nos fuerza a establecer un estricto orden temporal.

Al disponer de sólo una máquina el ensamblado final del objeto sólo se podrá realizar luego de N pasos de procesamiento o fabricación.

Si disponemos de N máquinas para fabricar el objeto, y no hay dependencias (x ej de la materia prima), cada una puede trabajar al mismo tiempo en una parte

Consecuencias.

-Menor tiempo para completar el trabajo

-Menor esfuerzo individual

-Paralelismo del hardware

Dificultades:

-Distribución de la carga de trabajo

-Necesidad de compartir recursos evitando conflictos

-Necesidad de esperarse en puntos clave

-Necesidad de comunicarse

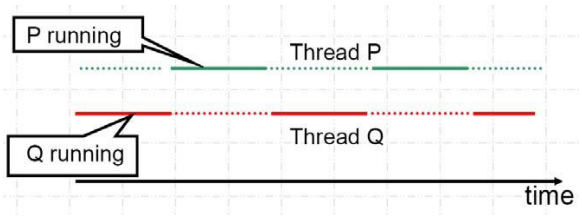
-Tratamiento de las fallas

-Asignación de una de las máquinas para el ensamblado (Cual??)

Vimos las soluciones secuencial y paralela (multiplicando el hard) en el problema de fabricar un objeto (sistema) de múltiples partes

Otro enfoque:

UNA máquina dedica parte del tiempo a cada componente del objeto=> Concurrencia sin paralelismo de hard



Dificultades.

-Distribución de carga de trabajo-Necesidad de compartir recursos evitando conflictos

-Necesidad de esperarse en puntos clave-Necesidad de comunicarse

-Necesidad de recuperar el "estado" de cada proceso al retomarlo.

CONCURRENCIA=>Concepto de software no restringido a una arquitectura particular de hardware ni a un número determinado de procesadores

PROCESOS:

Qué es un proceso?

PROCESO: programa secuencial

Un único thread o flujo de control

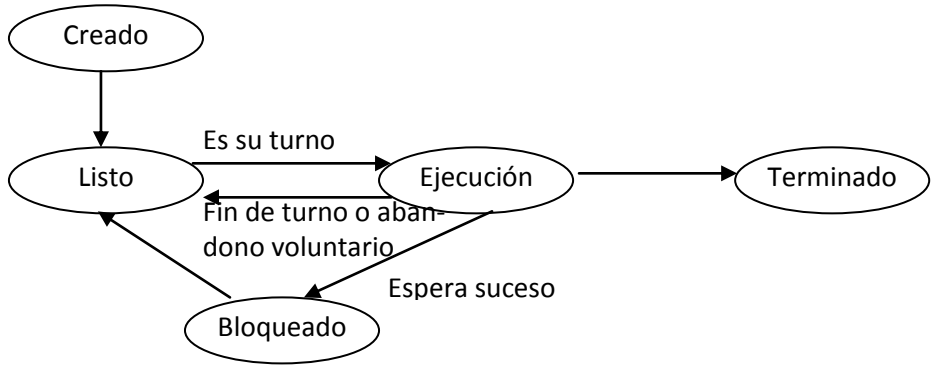
- programación secuencial, monoprocesador

Múltiples threads o flujos de control

- programa concurrente
- .procesos paralelos

Los procesos cooperan y compiten...

Ciclo de vida de un proceso: es prácticamente un estándar en todos los sistemas operativos. En principio el proceso no existe, es creado, luego pasa a listo (el proceso está en condiciones de usar la cpu) hasta que se el planificador de procesos scheduler (que suele ser parte del sistema operativo) le dé la oportunidad. de usar el procesador.
Los procesos tienen estado listo hasta que el planificador decide darles tiempo de ejecución pasando al estado en ejecución.



Un proceso puede pasar de ejecución a listo, esta decisión la toma el planificador. Que sigue alguna política para asignar la CPU a los distintos procesos. Una forma que lo realiza es mediante la asignación de un quantum de tiempo, de tal forma que cuando un procesador cumpla su tiempo de permanencia en el procesador, es desalojado pasando nuevamente a estado listo. Puede que un proceso abandone voluntariamente la cpu y pase a listo.
Un proceso puede pasar de estado en ejecución a bloqueado cuando ha de esperar que suceda un evento o suceso. Ejemplo: esperar un operación de entrada salida, la espera de finalización de una tarea por parte de otro proceso, un bloqueo voluntario durante un determinado tiempo, etc. Una vez ocurrido el evento que estaba esperando, el proceso pasa a listo. El acto de cambiar un proceso de estado se denomina **cambio de contexto.**

Clases de aplicaciones:
Programación Concurrente ⇒
- -organizar software que consta de partes (relativamente) independientes
- -usar uno o múltiples procesadores

3 grandes clases (superpuestas) de aplicaciones:
- -Sistemas multithreaded
- -Sistemas de cómputo distribuido
- -Sistemas de cómputo paralelo

Ejecución de N procesos independientes en M procesadores (N>M).
Un sistema de software de "multithreading"maneja simultáneamente tareas independientes, asignando los procesadores de acuerdo a alguna política (ej, por tiempos).
Organización más "natural" como un programa concurrente.

Ejemplos:
-Sistemas de ventanas en PCso WS
-Sistemas Operativos time-sharedy multiprocesador
-Sistemas de tiempo real (x ej, en plantas industriales o medicina)
Procesamiento paralelo
Resolver un problema en el menor tiempo (o un problema + grande en aprox. el mismo tiempo) usando una arq. multiprocesador en la que se pueda distribuir la tarea global en tareas (independientes? interdependientes?) que puedan ejecutarse en ≠procesadores.
Paralelismo de datos y paralelismo de procesos.

Ejemplos:
-Cálculo científico. Modelos de sistemas (meteorología, movimiento planetario, ...)
-Gráficos, procesamiento de imágenes, efectos especiales, procesamiento de video, realidad virtual
-Problemas combinatorios y de optimización lineal o no lineal. Modelos econométricos

Suma de 32 números
1 persona (secuencial): 32-1 sumas, c/u 1 unidad de tiempo
Entre dos: c/u suma 16. Reducción del tiempo a la mitad

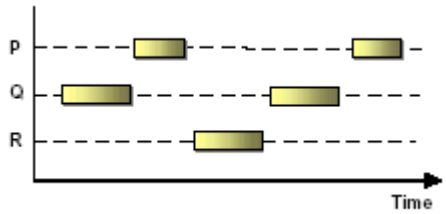
Siguiendo →árbol binario profundidad 5 →suma en 5 pasos y 25-1adiciones
Máxima concurrencia (16 personas en este caso) pasan los resultados a 23(8 de los 16), luego a 22, etc.
El resultado lo obtiene el último "sumador"
El cómputo concurrentecon distintos grados de concurrencia resulta en un menor tiempo de ejecución

El procesamiento paralelo lleva a los conceptos de speedupy eficiencia.

Speedup⇒S =Ts/ Tp
Qué significa??
Rango de valores (x qué está limitado?). Gráfica.

Eficiencia ⇒E = S / p
Qué representa??
Rango de valores. Gráfica.
En la ejecución concurrente, el "speedup"es menor

Programa Concurrente
Un proceso o tarea es un elemento concurrente abstracto que puede ejecutarse simultáneamente con otros procesos o tareas, si el hardware lo permite (x ejemplo TASKs de ADA).



Un programa concurrente puede tener N procesos habilitados para ejecutarse concurrentemente y un sistema concurrente puede disponer de M procesador es cada uno de los cuales puede ejecutar uno o más procesos. Un programa concurrente especifica dos o más programas secuenciales que pueden ejecutarse concurrentemente en el tiempo como tareas o procesos.

Programa secuencial ⇒
-**totalmente ordenado**
-**determinístico:** para los mismos datos de entrada, ejecuta siempre la misma secuencia de instrucciones y obtiene la misma salida

Esto no es verdad para los programas concurrentes...

Ejemplo⇒
- x=0; //P
- y=0; //Q
- z=0; //R

-En este caso, el orden de ejecución es irrelevante
-Tenerla computación distribuida en 3 máquinas sería más rápido
-Nota: hay instrucciones que requieren ejecución secuencial(x ej, x=1; x=x+1;)

Secuencialidad y concurrencia

Concurrencia lógica
Qué pasa si tenemos sólo 1 o 2 procesadores?
-qué instrucciones ejecutamos primero
-importa?
Son instrucciones que pueden ser lógicamente concurrentes

P →Q →R
Q →P →R
R →Q →P …

Darán el mismo resultado

Orden parcial
Las instrucciones pueden tener overlapping:

-si descomponemos P en p1, p2, ..., pn(también Q y R)
-podemos tener los ordenamientos
–p1,p2,q1,r1,q2…
–q1,r1,q2,p1…
•La única regla es que pi ejecuta antes que pj sii<j (orden parcial)
•No sabemos nada respecto del orden efectivo de ejecución (importa?)

En el caso anterior no nos preocupó el orden... cualquiera de las ejecuciones (con distinto orden) dará el mismo resultado
Pero en general esto no es así: diferentes ejecuciones, con la misma entrada, pueden dar distintos resultados
Ejemplo: Sup. que x es 5
•x=0; //P
•x=x+1; //Q

Escenario1	Escenario2
P →Q →x=1	Q →P →x=0

•Los programas concurrentes pueden ser no-determinísticos: pueden dar distintos resultados al ejecutarse sobre los mismos datos de entrada

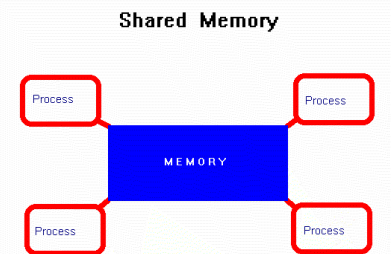
Conceptos básicos de concurrencia
Un programa concurrente puede ser ejecutado por:
Multiprogramación: los procesos comparten uno o más procesadores
Multiprocesamiento: cada proceso corre en su propio procesador pero con memoria compartida
Procesamiento Distribuido: cada proceso corre en su propio procesador conectado a los otros a través de una red
Tres características de los sistemas concurrentes:
Interacción:

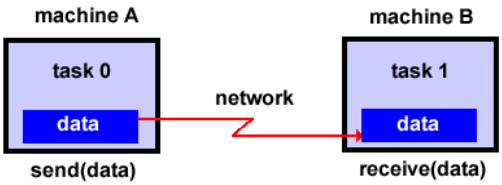
Los componentes interactúan unos con otros. Mecanismos.

Son "**no transformacionales**":

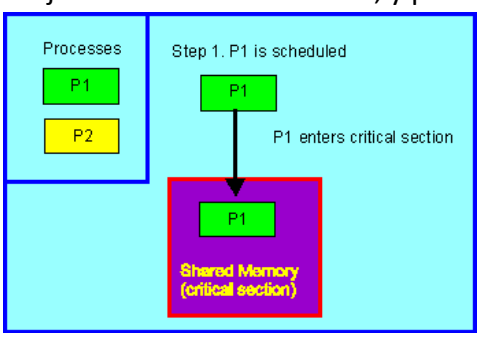
Los sistemas transformacionales pueden verse como transformadores de E/ en S/ (funciones).
En los sistemas concurrentes, los componentes evolucionan en paralelo con su entorno. No se definen funciones que caracterizan el comportamiento "de E/ a S/"sino en términos de las interacciones

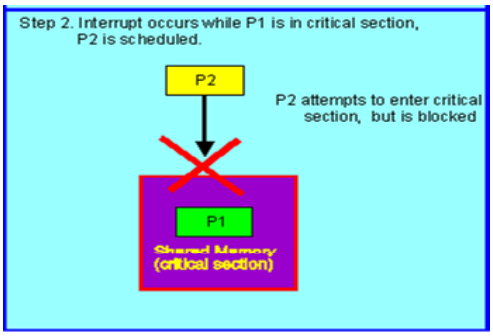
Ejecución ilimitada: Es inapropiado poner una cota superior a la ejecución. Ej: Internet
Comunicación de procesos en la concurrencia:
Los procesos se COMUNICAN-
Por Memoria Compartida -Por Pasaje de Mensajes





Los procesos se SINCRONIZAN por exclusión mutua en el acceso a secciones críticas de código para no ejecutar simultáneamente, y por condición





Ejemplo: sincronización para la reserva de pasajes

Sincronización

El problema de la reserva de pasajes…

- Travel agents might run the following code:

```
void reserveSeat(Position p) {
   if (seat.free(p))
      seat.reserve(p);
}
```

- and then issue a valid ticket for the seat at position p

--

Sincronización

Travel agent A

```
void reserveSeat(25J) {
   if (seat.free(25J))
      seat.reserve(25J);
}
```

Travel agent A

Travel agent B

```
void reserveSeat(25J) {
   if (seat.free(25J))
      seat.reserve(25J);
}
```

Travel agent B

Ejemplos de sincronización.

-Completar las escrituras antes de que comience una lectura

-Cajero: dar el dinero sólo luego de haber verificado la tarjeta
-Compilar una clase antes de reanudar la ejecución
-No esperar por indefinidamente, si la otra parte está “muerta”

Sincronización
Sincronización=> posesión de información acerca de otro proceso para coordinar actividades.
Estado de un programa concurrente.
Cada proceso ejecuta un conjunto de sentencias, cada una implementada por una o más acciones atómicas (indivisibles).
Historia de un programa concurrente.
Se debe asegurar un orden temporal entre las acciones que ejecutan los procesos.
Las tareas se intercalan en el tiempo .deben fijarse restricciones

El objetivo de la sincronización es restringir las historias de un programa concurrente sólo a las permitidas

Formas de sincronización:
Sincronización por exclusión mutua
Asegurar 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, EM evita que dos o más procesos puedan encontrarse en la misma sección crítica al mismo tiempo.

Sincronización por condición:
Permite bloquear la ejecución de un proceso hasta que se cumpla una condición dada.
Ejemplo de los dos mecanismos de sincronización en un problema de utilización de un área de memoria compartida (buffer limitado con productores y consumidores)

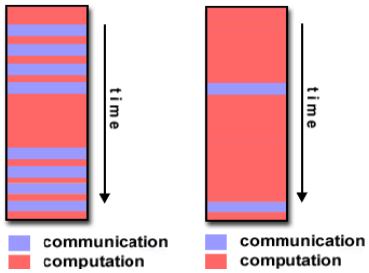
Comunicación y Prioridad:
La comunicación entre procesos concurrentes indica el modo en que se organiza y trasmiten datos entre tareas concurrentes.
Esta organización requiere especificar protocolos para controlar el progreso y corrección de la comunicación.
Los protocolos deben contemplar la posibilidad de pérdida de información.
Un proceso que tiene mayor prioridad puede causar la suspensión (pre-emption) de otro proceso concurrente.

Análogamente puede tomar un recurso compartido, obligando a retirarse a otro proceso que lo tenga en un instante dado.

Conceptos relacionados con PC: Granularidad
Elección de la Granularidad.

- Grano fino
- grano grueso

Para una aplicación dada, significa optimizar la relación entre el número de procesadores y el tamaño de memoria total.
Puede verse también como la relación entre cómputo y comunicación



Conceptos relacionados con PC: Manejo de los recursos

Manejo de los recursos.

Uno de los temas principales de la programación concurrente es la administración de recursos compartidos.

Esto incluye la asignación de recursos compartidos, métodos de acceso a los recursos, bloqueo y liberación de recursos, seguridad y consistencia.

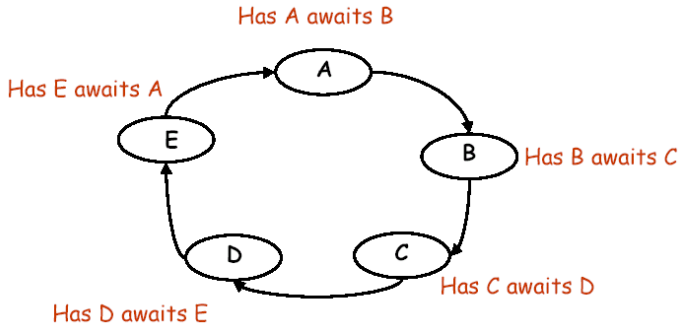
Una propiedad deseable en sistemas concurrentes es el equilibrio en el acceso a recursos compartidos por todos los procesos (fairness).

Dos situaciones NO deseadas en los programas concurrentes son la inanición de un proceso (no logra acceder a los recursos compartidos) y el overloading de un proceso (la carga asignada excede su capacidad de procesamiento).

Conceptos relacionados con PC: El problema del deadlock

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

La ausencia de deadlock es una propiedad necesaria en los procesos concurrentes.



4 propiedades necesarias y suficientes para que exista deadlock:

-Recursos reusables serialmente

-Los procesos comparten recursos que pueden usar con EM (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, c/u tiene un recurso que su sucesor en el ciclo está esperando adquirir

Algunos comentarios

Existe un no determinismo implícito en el interleaving de procesos concurrentes. Esto significa que dos ejecuciones del mismo programa no necesariamente son idénticas=>dificultad para la interpretación y debug.
Posible reducción de performance por overhead de context switch, comunicación, sincronización, ...
Aumenta el tiempo de desarrollo y puesta a punto respecto de los programas secuenciales, y puede aumentar el costo de los errores
La paralelización de algoritmos secuenciales no es un proceso directo, que resulte fácil de automatizar.
Para obtener una mejora real de performance, se requiere adaptar el software concurrente al hardware paralelo (mapeo)

Mecanismos de comunicación y sincronización entre procesos
Memoria compartida (MC)
Los procesos intercambian información sobre la memoria compartida o actúan coordinadamente sobre datos residentes en ella.
Lógicamente no pueden operar simultáneamente sobre la MC, 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 MC.

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.

Requerimientos para un lenguaje concurrente
“Independientemente del mecanismo de comunicación / sincronización entre procesos, los lenguajes de programación concurrente deberán proveer primitivas adecuadas para la especificación e implementación de las mismas.”
De un lenguaje de programación concurrente se requiere:
- Indicar las tareas o procesos que pueden ejecutarse concurrentemente.
- Mecanismos de sincronización
- Mecanismos de comunicación entre los procesos.

Resumen de conceptos
La Concurrencia es un concepto de software.
La Programación Paralela se asocia con la ejecución concurrente en múltiples procesadores que pueden tener memoria compartida, y generalmente con un objetivo de incrementar performance.
La Programación Distribuida es un “caso” de concurrencia con múltiples procesadores y sin memoria compartida.
En Programación Concurrente la organización de procesos y procesadores constituyen la arquitectura del sistema concurrente.

“Especificar la concurrencia es esencialmente especificar los procesos concurrentes, su comunicación y sincronización.”

Tareas propuestas

Leer los capítulos 1 y 2 del libro de Andrews
Leer los ejemplos de paralelismo recursivo, productores y consumidores, clientes y servidores y el código de la multiplicación de matrices distribuida del Capítulo 1 (Andrews).
Investigar las primitivas de programación concurrente de algún lenguaje de programación.