# UD 2: PROGRAMACIÓN CONCURRENTE Y SISTEMAS DE TIEMPO REAL

TEMA 5: INTRODUCCIÓN A LA PROGRAMACIÓN CONCURRENTE. PROCESOS E HILOS. EL PLANIFICADOR DEL SISTEMA OPERATIVO. POSIX

« Me lo explicaron y lo olvidé, lo ví y lo aprendí, lo hice y lo entendí.»

- Confucio





## INTRODUCCIÓN

"Coincidencia, concurso simultáneo de varias circunstancias" RAE

- La programación concurrente nació en el desarrollo de los SOs:
  - Se debía repartir el tiempo de procesador entre varios usuarios, que además podían ejecutar múltiples programas
  - Optimizar el uso de la CPU: las operaciones de E/S son muy lentas comparadas con su velocidad
  - Se programaba a muy bajo nivel de abstracción: ensamblador
- A partir de los '80 se extendió a los programas de usuario:
  - Lenguajes concurrentes
  - Librerías de gestión de la concurrencia: POSIX
- Fuertemente condicionada por el hardware disponible
  - Sistemas multi-procesador con/sin memoria compartida
  - Conjunto de instrucciones del procesador

### **HECHOS CONSTATADOS**

- La **ley de Moore** expresa que aproximadamente cada dos años se duplica el número de transistores en un circuito integrado
- La **ley de Amdahl** establece que 'n' procesadores funcionando en paralelo a velocidad 'm' no tienen la misma potencia de cómputo que 1 procesador a velocidad 'n\*m', debido a las dependencias que existen entre los programas que ejecutan
- La variación lineal en la frecuencia de la CPU produce variación lineal del rendimiento pero exponencial del consumo (y generación de calor)
- Límite térmico del silicio entorno a 4GHz (el físico se encuentra a 10Ghz, con nitrógeno líquido se han alcanzado 9GHz)
- ➤ La tecnología avanza hacia sistemas manycore y GPUs → sistemas tremendamente paralelos con HW dedicado

### BENEFICIOS

Aumento del rendimiento En entornos monoprocesador



En aplicaciones con procesos intensivos en E/S

En entornos multiprocesador Posibilidad de asignar procesos a diferentes procesadores

Cuando existen "algoritmos concurrentes" más eficientes

Aumenta la capacidad de modelado: permite diseñar los programas de acuerdo con la realidad



Da solución fácil y más elegante a problemas que son inherentemente concurrentes

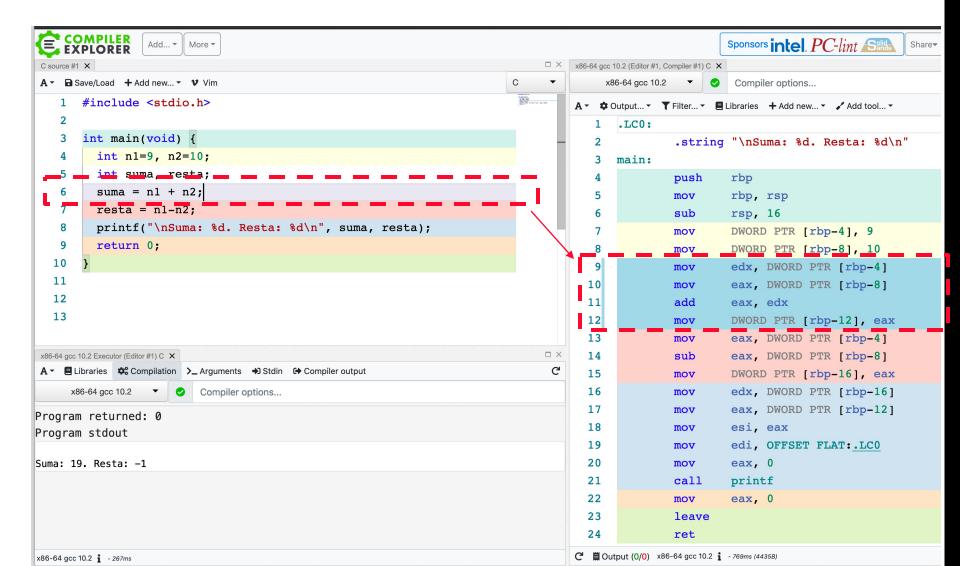
En general, mejora los tiempos de respuesta y el rendimiento de las aplicaciones

### PROGRAMA CONCURRENTE

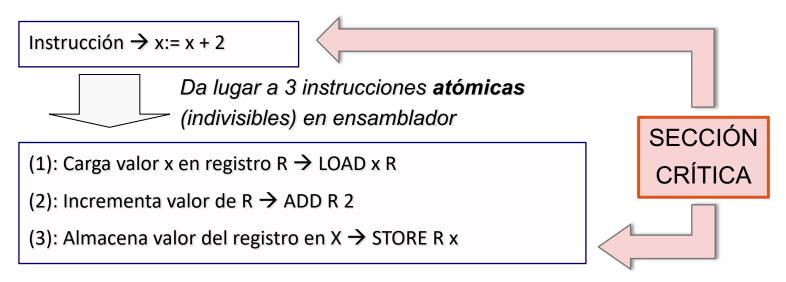
Un programa concurrente está formado por un conjunto finito de procesos secuenciales

- ➤ Los procesos secuenciales están escritos usando un conjunto de sentencias atómicas (indivisibles) → instrucciones ensamblador
- La ejecución de un programa concurrente consiste en la ejecución de una secuencia de instrucciones atómicas resultante del intercalado arbitrario de las instrucciones atómicas de los procesos que lo forman
- Existe indeterminismo en la ejecución de programas concurrentes
- Los efectos de las acciones de un proceso se pueden hacer visibles al resto de procesos secuenciales en cualquier orden

### **EJEMPLO "INOFENSIVO"**



# EXCLUSIÓN MUTUA, SECCIÓN CRÍTICA Y CONDICIONES DE CARRERA



Supongamos que x puede ser incrementada concurrentemente por H1 y H2

Х	0	0	0	0	2	2	2
H1	(1)	(2)			(3)		
H2			(1)	(2)		(3)	

¡Se pierde un incremento!

Condición de carrera: cuando el resultado final depende del orden en que se intercalan las instrucciones de dos o más hilos

# PROBLEMAS DE LA PROGRAMACIÓN CONCURRENTE

- Exclusión mutua: dos o más procesos no deben acceder simultáneamente a variables compartidas
  - El código que debe ejecutarse en régimen de exclusión mutua se denomina sección crítica
  - Las secciones críticas deben ejecutarse en exclusión mutua, es decir, sólo un proceso debe estar en la sección en un instante dado
- Sincronización: hay situaciones en las que un proceso no puede avanzar hasta que otro proceso haya realizado alguna acción
- Los sistemas operativos proporcionan mecanismos para solucionar ambos problemas

### **PROBLEMAS REALES**

- Es fácil cometer errores, sobre todo al principio
- Se necesita ser muy riguroso y cuidadoso y tener paciencia
  - Therac 25 Máquina de terapia radioactiva
    - Errores en la programación concurrente contribuyeron a la ocurrencia de accidentes que causaron muertes y heridos graves
    - Leveson, Nancy, and Clark S. Turner. July 1993. "An investigation of the Therac-25 accidents" *IEEE Computer*, Vol. 25, No. 7, pp. 18-41.
  - Mars Rover
    - Problemas con la interacción entre tareas concurrentes provocaron resets periódicos que comprometieron gravemente la disponibilidad del robot www.cs.cornell.edu/cs614-sp99/papers/pathfinder.html

### PROCESOS E HILOS

- > En los primeros SOs el proceso era:
  - La unidad de propiedad de recursos: espacio de direcciones, canales de E/S, ficheros, etc.
  - La unidad de expedición (o planificación): a cada proceso se le asigna la CPU durante un tiempo
- Puesto que ambos aspectos pueden gestionarse por separado, en los SOs modernos:
  - Proceso (process o heavyweight process): unidad de propiedad de recursos
  - Hilo (thread o lightweight process): unidad de expedición. Parte realmente concurrente del programa

### PROCESS CONTROL BLOCK (PCB)

- Guarda la información relativa a un proceso: estado del proceso + control del proceso
- Información utilizada por el planificador y el SO
- En Linux, sobre 95 campos

#### **BLOQUE DE CONTROL DEL PROCESO**

#### Identificación del proceso

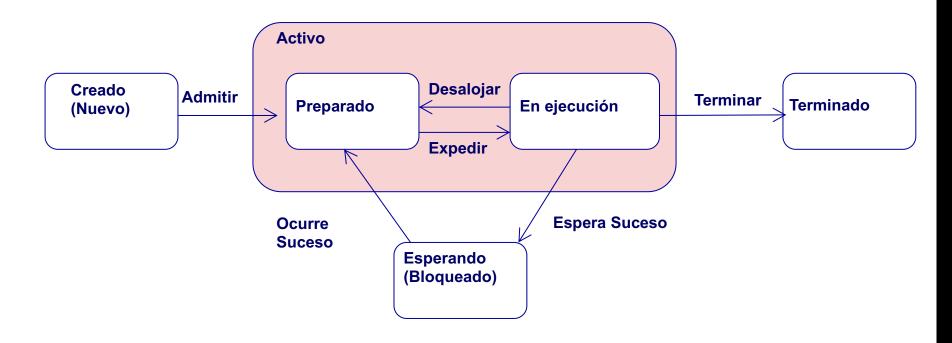
#### Información de estado del proceso:

- Estado de ejecución (listo, bloqueado, ...)
- Contador de programa
- Registros de la CPU
- Punteros de pilas
- Código estático
- Etc,...

#### Información de control del proceso:

- Gestión de memoria: espacio asignado
- Recursos asignados (ficheros, sockets, etc.) y estado
- Información de contabilidad
- Privilegios y planificación
- Etc,...

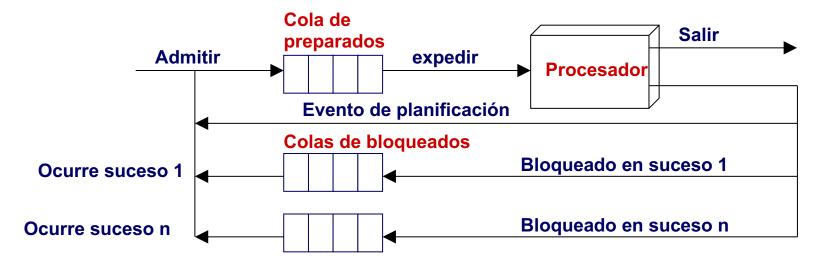
### CICLO DE VIDA PROCESO/HILO



Bajo el control del planificador

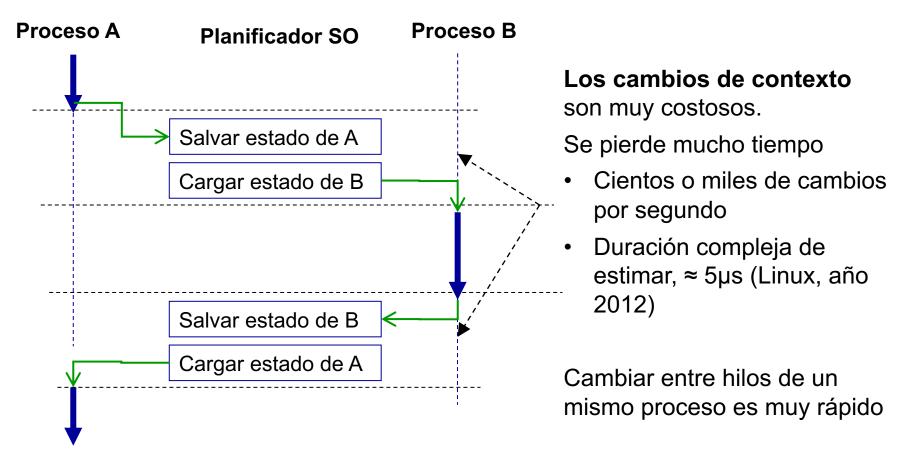
## PLANIFICADOR (SCHEDULER)

- Programa que forma parte del SO y que decide qué proceso/hilo ejecutar a continuación, realizando el cambio de contexto
- El planificador se ejecuta al recibir un evento de planificación:
  - Llamadas al sistema de E/S (teclado, fichero, memoria, socket, etc.)
  - o Bloqueo voluntario, acceso a recursos compartidos, sincronización
  - Y, dependiendo del tipo de planificador:
    - Fin de la ejecución del proceso actual
    - Fin del tiempo máximo de ejecución (quantum)
    - Abandono voluntario (yield)
    - Llegada a la cola de preparados de un proceso con mayor prioridad



### **CAMBIO DE CONTEXTO**

Cada vez que en la CPU se cambia de proceso/hilo es necesario guardar el estado del antiguo y cargar el del nuevo



### **EL STANDARD POSIX**

- Portable Operating System Interface
  - Diseñado para el sistema operativo UNIX
- Estándar desarrollado conjuntamente por la Computer Society del IEEE y The Open Group
  - también denominado The Single UNIX Specification
  - la denominación oficial es IEEE Std. 1003.1, e ISO/IEC-9945
- Participan fabricantes de hardware, proveedores de SOs, herramientas de desarrollo de software, diseñadores de software, académicos, programadores, etc.

http://www.unix.org/online.html

# DOS ENFOQUES A LA PROGRAMACIÓN CONCURRENTE

- Enfoque tradicional para sistemas de tiempo-real (comportamiento analizable y predecible) o con recursos limitados: el programador crea y gestiona hilos y variables compartidas
  - Thread, mutex, variable de condición, etc.
- Enfoque para sistemas many-core: el programador marca en el código qué partes pueden ser ejecutadas de forma paralela y el sistema reparte la carga automáticamente entre los núcleos disponibles
  - Tareas asíncronas, futuros, etc.
- No son excluyentes

## ¡EL ORDENADOR NO EJECUTA EL CÓDIGO QUE ESCRIBIMOS!

- El compilador, el procesador y la memoria lo "modifican":
  - Optimizaciones, reducción de expresiones, reordenamiento de lecturas/escrituras de variables, etc.
  - Predicción de saltos, cachés locales, etc.
  - Buffers de memoria, lectura predictiva, lecturas/escrituras selectivas, etc.
- Estas modificaciones afectan solo al rendimiento en programas con un solo hilo, pero pueden afectar al funcionamiento de programas concurrentes si no están bien diseñados
- Siempre hay que medir: *Early optimization is the root of all evil*http://channel9.msdn.com/Shows/Going+Deep/Cpp-and-Beyond-2012Herb-Sutter-atomic-Weapons-1-of-2

### **MODELO DE MEMORIA EN C/C++11**

Un "modelo de memoria" describe cómo se realizan las interacciones entre hilos a través de memoria compartida, ya que el compilador desconoce qué es un hilo (solo invoca funciones)

Modelo C/C++: consistencia secuencial para programas libres de condiciones de carrera (desde 2011, igual que en Java)

Si el programador hace un programa libre de condiciones de carrera, el compilador genera un código que se comporta como el desarrollado (visto como una caja negra)

Es imprescindible identificar en el código y proteger adecuadamente el acceso a variables compartidas entre hilos para asegurar que el compilador genera el código correcto

https://es.cppreference.com/w/cpp/language/memory model