Fundamentos de los Sistemas Operativos (FSO)

Departamento de Informática de Sistemas y Computadoras (DISCA)
Universitat Politècnica de València

Bloque Temático 2: Gestión de procesos

Unidad Temática 5

Hilos de Ejecución

fSO

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DISCA

- **Objetivos**
  - Introducir el concepto de **programación concurrente**
  - Introducir el concepto de **hilo de ejecución** y sus diferentes modelos de implementación
  - Estudiar las diferencias entre hilo de ejecución y proceso pesado
  - Examinar la problemática asociada al **uso compartido de memoria por** parte de **actividades concurrentes**

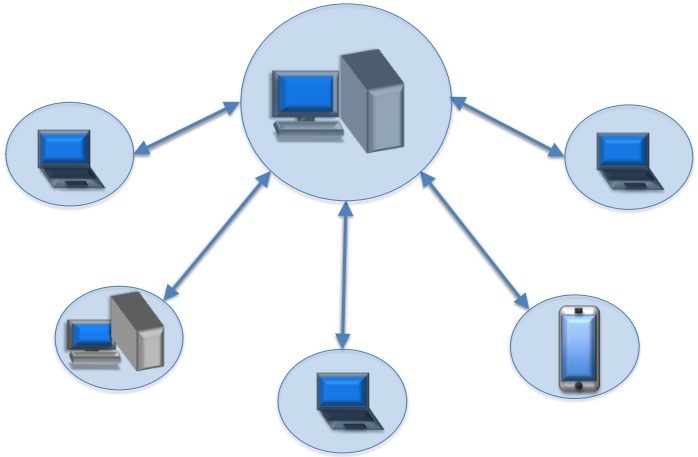
Bibliografía y contenido

- Contenido
  - Programación concurrente
  - Concepto de Hilo de ejecución
    - Proceso vs. Hilo de ejecución
  - Modelos de Hilos de ejecución
  - Necesidad de sincronización
  - Concepto de *Condición de Carrera*

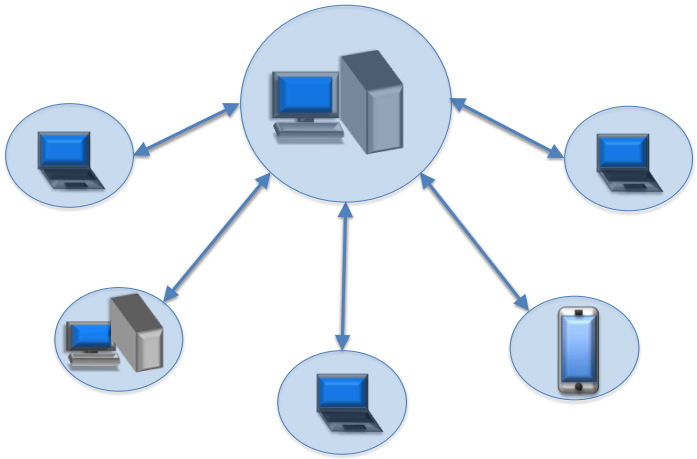
- Bibliografía
  - “Fundamentos de sistemas operativos” Silberschatz 7ª Ed.
  - “Sistemas operativos: una visión aplicada” Carretero 2º Ed.

- **Programación concurrente**
- Concepto de Hilo de ejecución
- Modelos de Hilos de ejecución
- Necesidad de sincronización
- Concepto de *Condición de Carrera*

- **Programación concurrente:**
  - Un **único programa** que intenta resolver un problema definiendo **varias "actividades"**
  - Existe un **paralelismo** potencial **entre tareas**
- **Ejemplos de aplicaciones** :
  - **Servidor web** capaz de atender más de una petición de cliente, de manera que cada petición es atendida por una *actividad*
  - Juegos de ordenador/consola en los que cada personaje/objeto representa una *actividad*



- Las “actividades” de un programa concurrente
  - trabajan en común y requieren
    - » **comunicarse** entre sí para intercambiar datos (a través de memoria compartida y/o paso de mensajes.)
    - » **sincronizar** sus líneas de flujo de control**.**
- Para implementar estas actividades dos posibilidades
  - Actividad = **Proceso**
  - Actividad = **Hilo de ejecución (“thread”)**

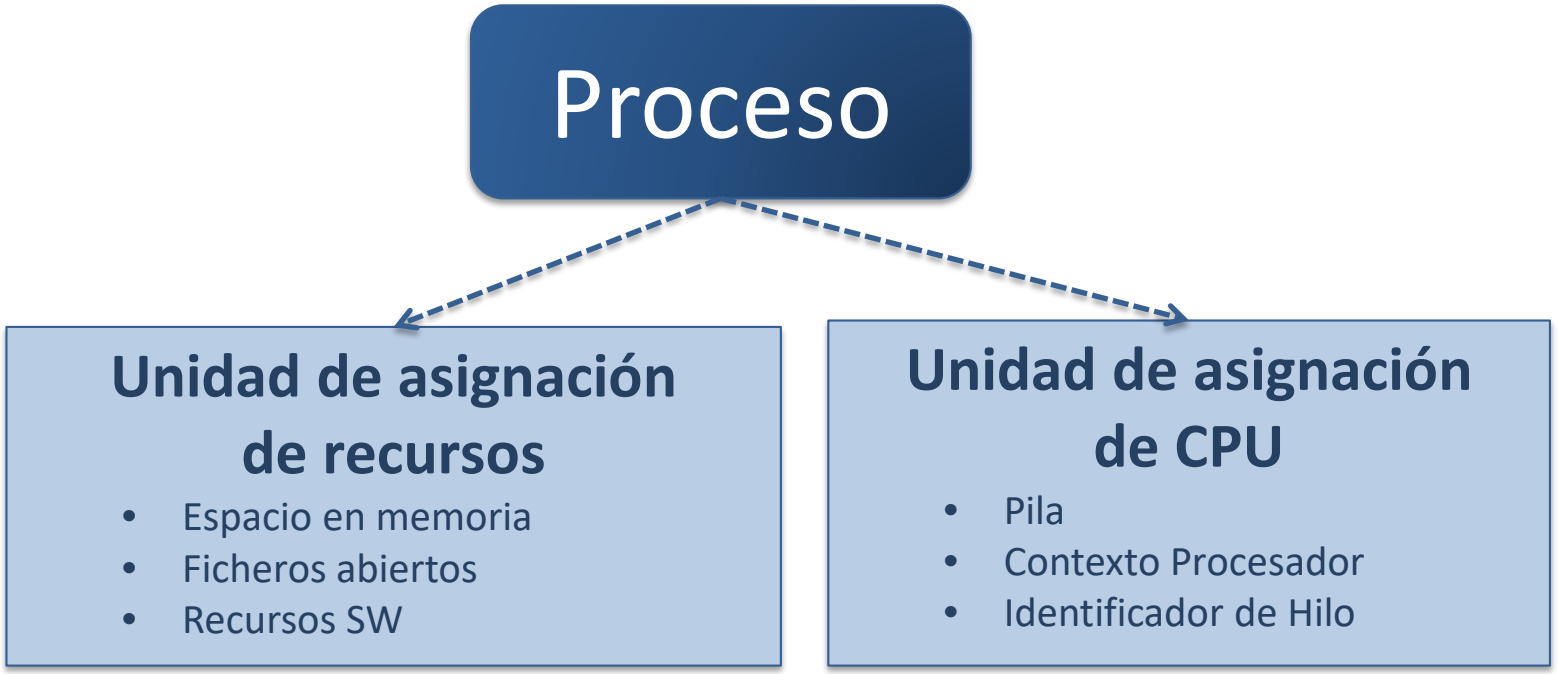


Contenido

- Programación concurrente
- **Concepto de Hilo de ejecución**
- Modelos de hilos de ejecución
- Necesidad de sincronización
- Concepto de *Condición de Carrera*

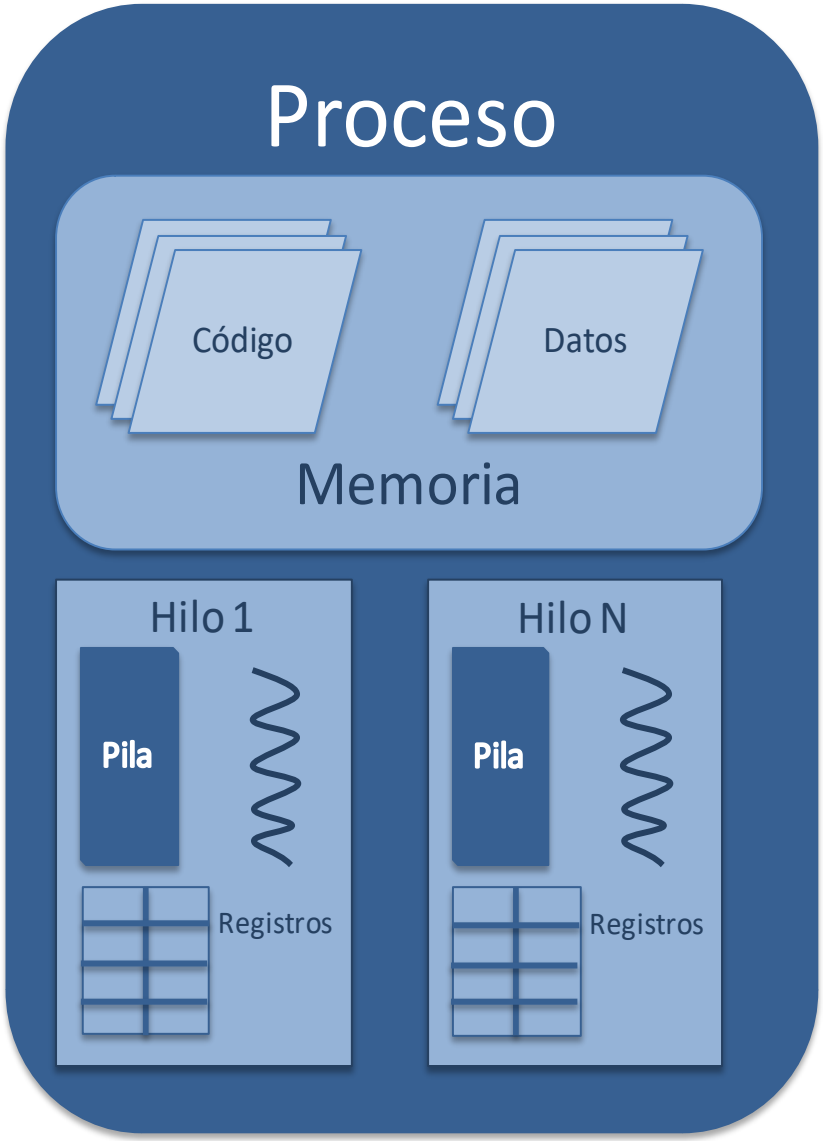
Concepto de hilo de ejecución

- Un proceso es una **entidad de abstracción** compuesta por:



- El Sistema Operativo puede disociar estas dos unidades de asignación del proceso

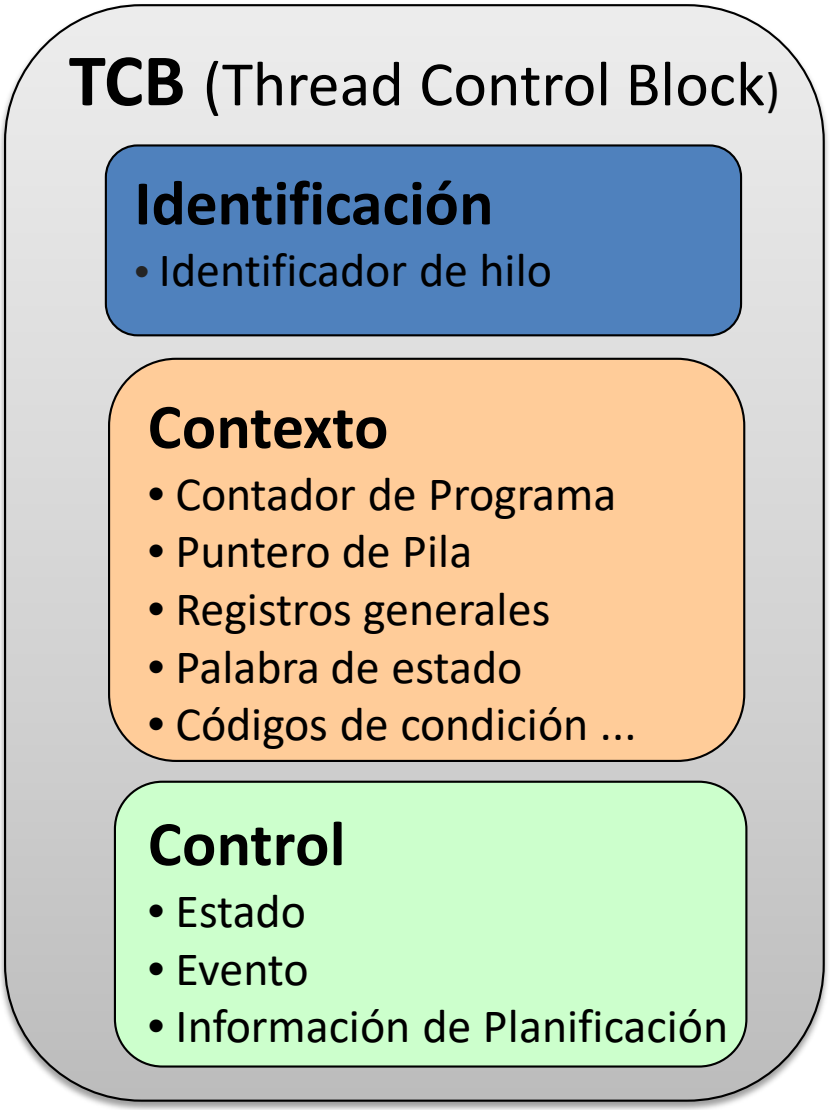
Concepto de hilo de ejecución



- **Hilo de ejecución:** Unidad básica de asignación de CPU.
- **Proceso** = Unidad de asignación de recursos con al menos un hilo de ejecución
- Los hilos de ejecución definidos **dentro de un mismo proceso** comparten:
  - Código
  - Datos
  - Recursos asignados al proceso
- Cada hilo dispone de **atributos propios**:
  - Identificador (ID)
  - Pila
  - Contador de programa
  - Registros

Concepto de hilo de ejecución

- Implementación de Hilos de Ejecución



- **Atributos**
  - Los hilos tienen **pocos atributos**
  - La información necesaria para soportar hilos de ejecución es más reducida que la que se debe mantener para los procesos pesados
  - La información de los recursos compartidos se guarda en el PCB del proceso

Concepto de hilo de ejecución

- **Proceso versus hilo**
  - Desde el **punto de vista del sistema**, un hilo es más barato
    - Cuesta menos...
      - » crear un hilo en un proceso existente que crear un proceso nuevo
      - » terminar un hilo que un proceso.
      - » cambiar de contexto entre dos hilos de un mismo proceso que entre dos procesos
  - Desde el **punto de vista del programador**, es más natural
    - Los hilos presentan un modelo de programación concurrente más sencillo, en el que la comunicación es:
      - » Más natural (los hilos comparten memoria/ficheros por definición)
      - » Más eficiente (en muchas ocasiones, no hace falta solicitar servicios al núcleo)

Contenido

- Programación concurrente
- Concepto de Hilo de ejecución
- **Modelos de hilos de ejecución**
- Necesidad de sincronización
- Concepto de *Condición de Carrera*

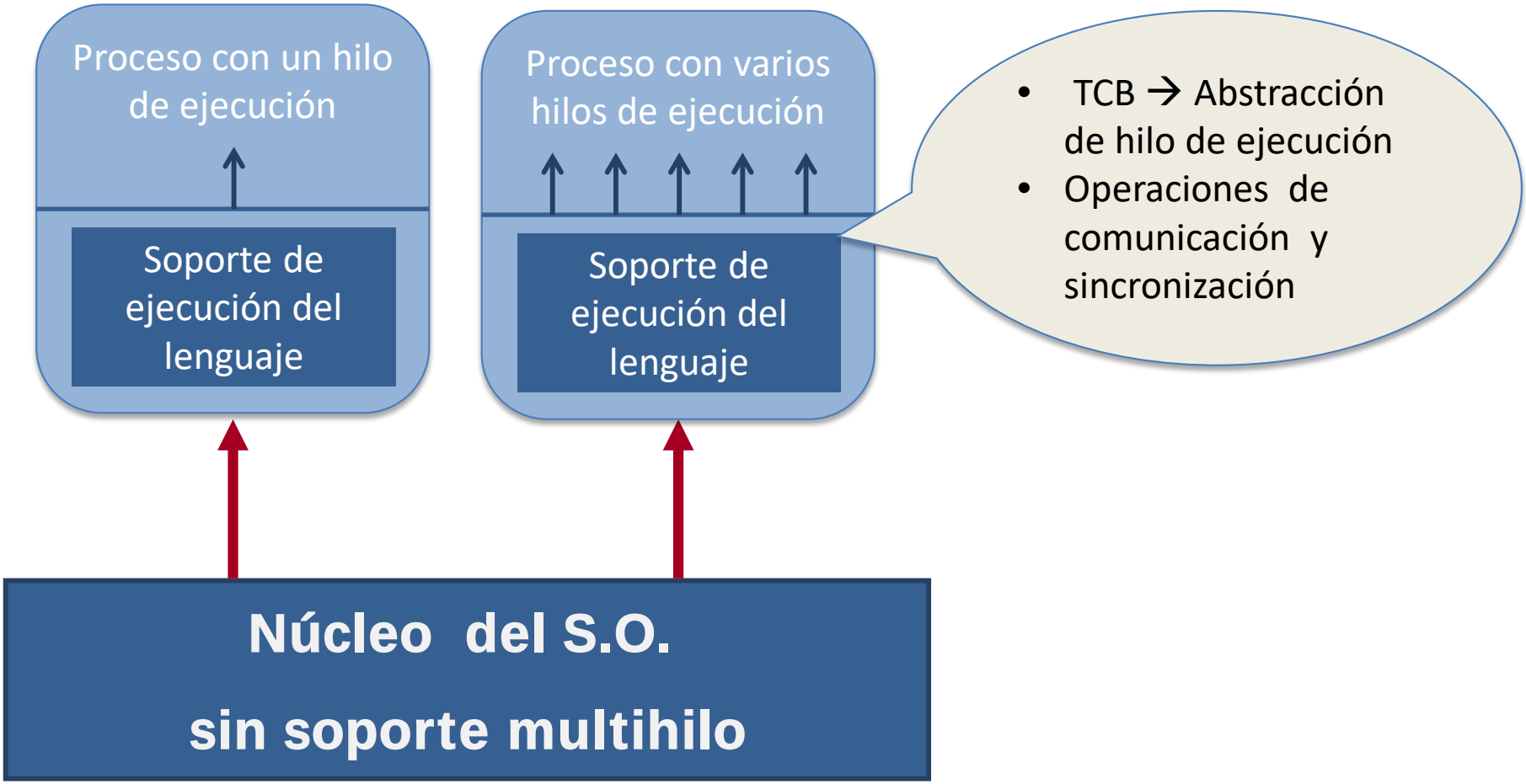
Modelos de hilos de ejecución

- Para programar con hilos es necesario **soportar** la abstracción de **hilo de ejecución**, en lo referente a:
  - Estructuras de datos con atributos de los hilos (TCBs)
  - Operaciones de comunicación y de sincronización de hilos

- En función de quien ofrece estas abstracciones, podemos encontrarnos con **tres modelos** de programación
  - Hilos a **nivel de usuario**
  - Hilos a **nivel de núcleo**
  - Hilos **híbridos**

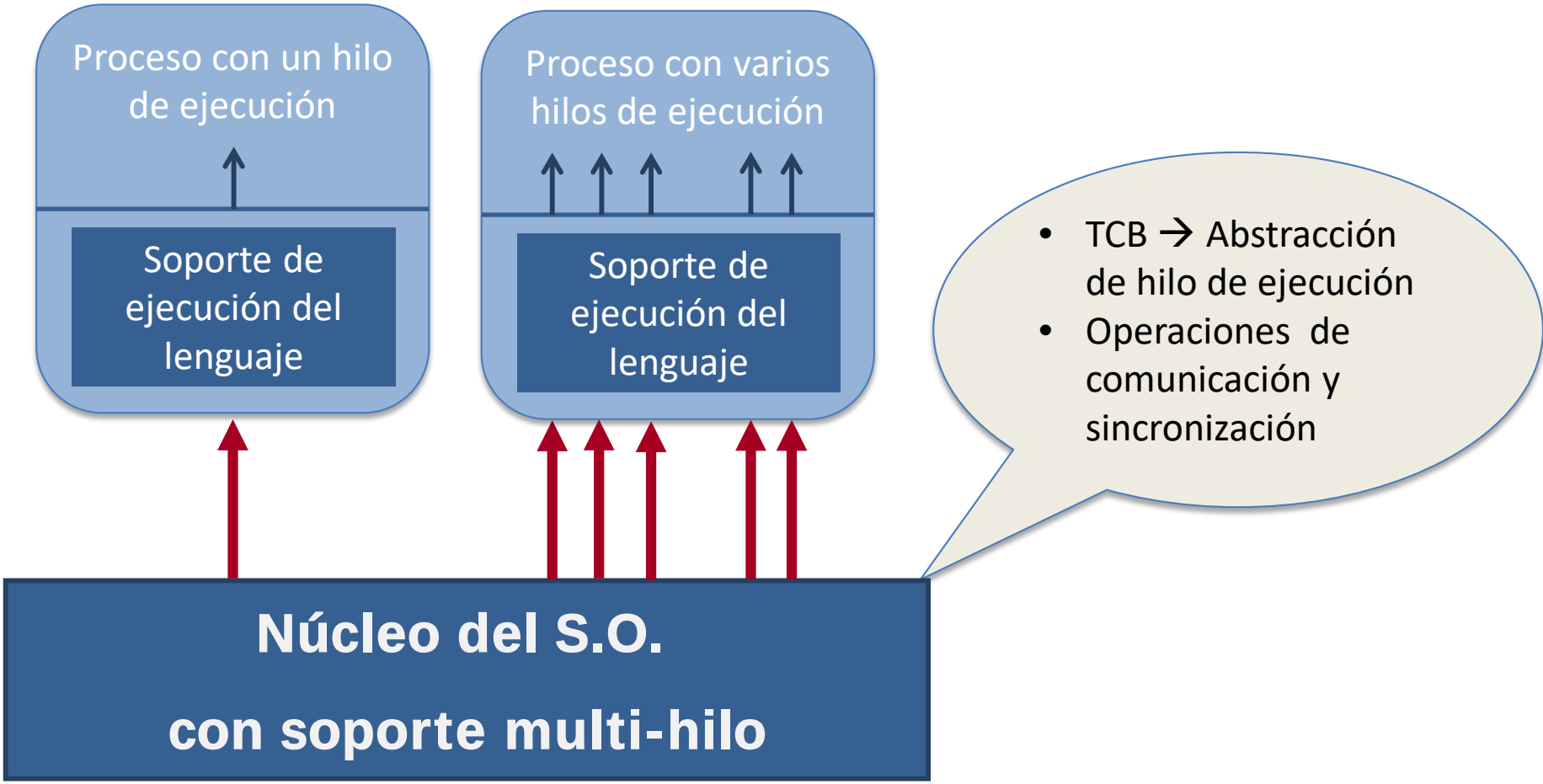
Modelos de hilos de ejecución

- **Hilos a nivel de usuario**
  - Las abstracciones las ofrece el soporte de ejecución del **lenguaje de programación** (“runtime”)



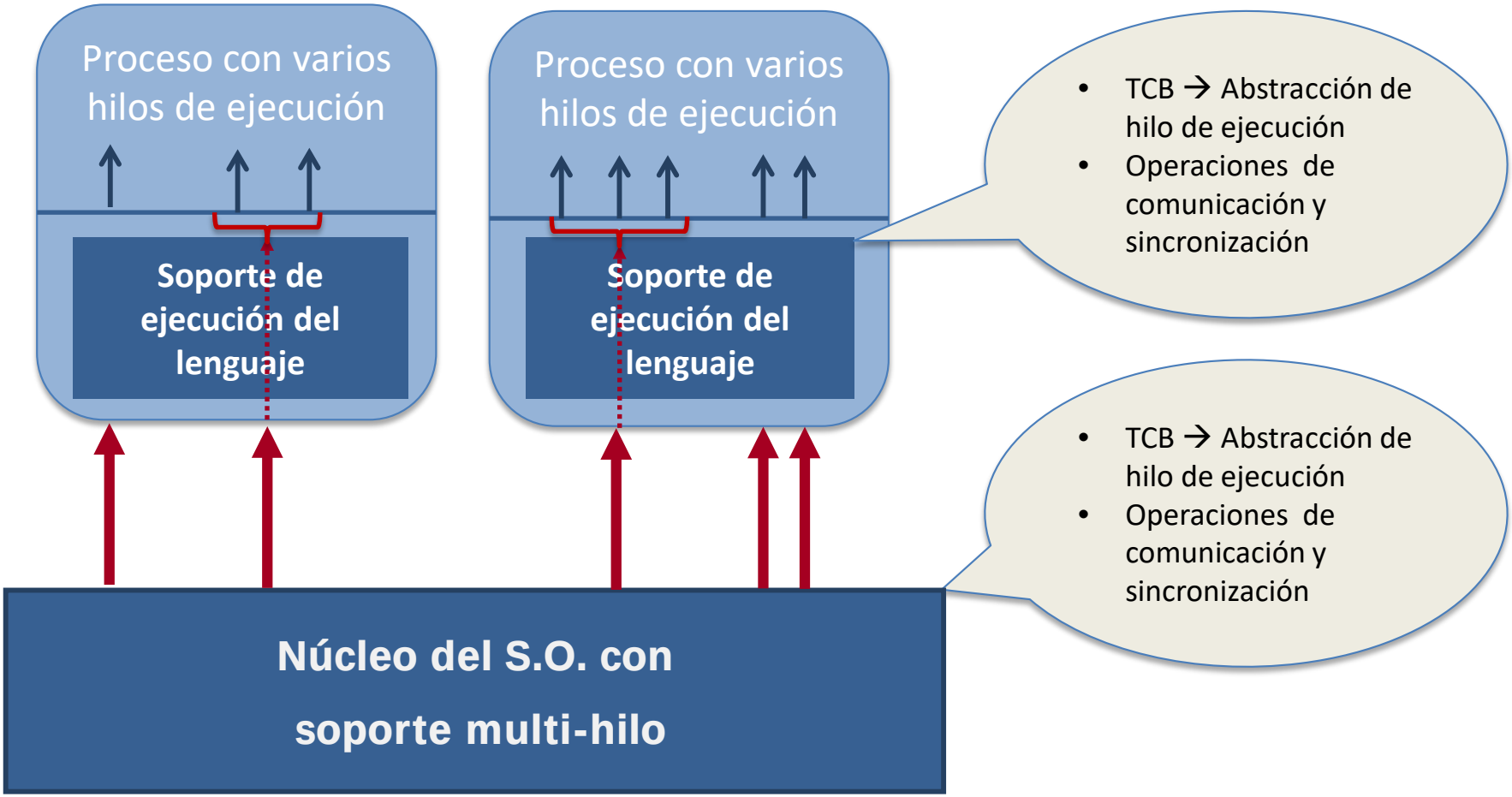
Modelos de hilos de ejecución

- **Hilos a nivel de núcleo**
  - Las abstracciones las ofrece el **núcleo** del sistema operativo mediante la interfaz de llamadas al sistema



Modelos de hilos de ejecución

- **Modelo Híbrido**
  - Las abstracciones las ofrecen el núcleo del sistema operativo y el soporte de ejecución del **lenguaje de programación** (“runtime”)



Contenido

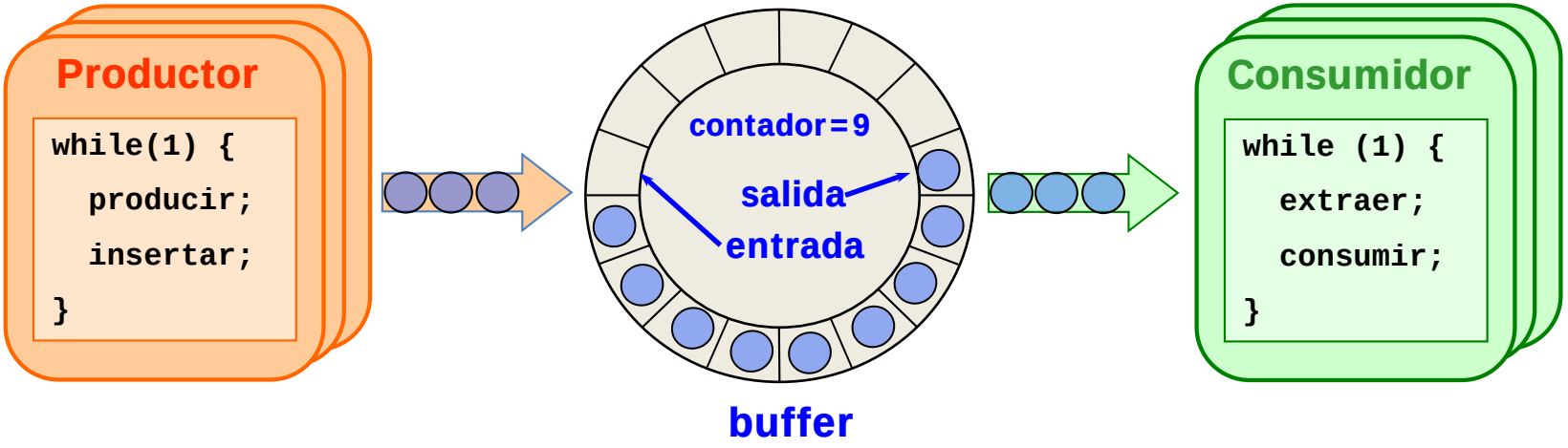
- Programación concurrente
- Concepto de Hilo de ejecución
- Modelos de hilos de ejecución
- **Necesidad de sincronización**
- Concepto de *Condición de Carrera*

Necesidad de Sincronización

- La **concurrencia es fundamental**
  - Tanto para las aplicaciones de usuario como en la estructura interna del Sistema Operativo se utiliza concurrencia.
- La programación concurrente aborda los siguientes aspectos:
  - Comunicación entre procesos
  - Compartición de recursos
  - Sincronización de actividades
  - Reserva de tiempo de CPU
- La concurrencia **se manifiesta tanto**
  - en **entornos de multiprocesadores** o distribuidos
  - como en **entornos monoprocesadores** y de tiempo compartido
- Se pueden distinguir tres contextos de concurrencia:
  - Múltiples aplicaciones
  - Una sola aplicación que se estructura en varias actividades (varios hilos o varios procesos)
  - Estructura del Sistema Operativo. El SO está implementado en forma de varias actividades

Necesidad de sincronización

- Ejemplo: problema del **"productor/consumidor"** ("buffer acotado")
  - Existen dos tipos de entidades: productores y consumidores (de "items")
  - Existe un buffer acotado (circular) que acomoda la diferencia de velocidad entre productores y consumidores:
    - Si el buffer se llena, los productores deben suspenderse
    - Si el buffer se vacía, los consumidores deben suspenderse



Necesidad de sincronización

- Código de hilos productor y consumidor

Comparten los hilos productores y consumidores

```
#define N  20
int  buffer[N];
int  entrada=0, salida=0, contador=0;
```

Bucles de **"espera activa"**

```
void *func_prod(void *p) {
  int item;

  while(1) {
    item = producir();

    while (contador == N)
      /*bucle vacio*/ ;
    buffer[entrada] = item;
    entrada = (entrada + 1) % N;
    contador = contador + 1;
  }
}
```

```
void *func_cons(void *p) {
  int item;

  while(1) {
    while (contador == 0)
      /*bucle vacio*/ ;
    item = buffer[salida];
    salida = (salida + 1) % N;
    contador = contador - 1;

    consumir(item);
  }
}
```

En este código:

"`contador`" y "`buffer`" son compartidos por el hilo productor y consumidor

Con varios hilos productores y consumidores, "`entrada`" sería compartida por todos los productores, y "`salida`" por todos los consumidores

Necesidad de sincronización

- **Productor/Consumidor,**
  - Hilos productores y consumidores se ejecutan de forma **concurrente**
    - accediendo a variables compartidas
  - Los hilos son elegidos para su **ejecución independientemente**
  - Las decisiones sobre **qué hilo se ejecuta** en cada momento se toman en cada **cambio de contexto. Dependen de un planificador** y no del programador de la aplicación.

- Esto puede producir que "*un código que es correcto si se ejecuta de forma secuencial, puede dejar de serlo cuando se ejecuta concurrentemente*", debido a una situación conocida como **condición de carrera**

Contenido

- Programación concurrente
- Concepto de Hilo de ejecución
- Modelos de hilos de ejecución
- Necesidad de sincronización
- **Concepto de *Condición de Carrera***

Concepto de Condición de Carrera

- Si suponemos que:
  - Inicialmente "contador" vale 5
  - Un productor ejecuta "contador = contador + 1;"
  - Un consumidor ejecuta "contador = contador - 1;"

el resultado final del "contador" debería ser 5

Productor:

```
contador = contador + 1;
```

```
lw reg1, contador
addi reg1, reg1, 1
sw reg1, contador
```

Consumidor:

```
contador = contador - 1;
```

```
lw reg2, contador
addi reg2, reg2, -1
sw reg2, contador
```

Pero si se ejecuta la siguiente secuencia de operaciones…

Cambios de contexto

T	Hilo	Operación	reg1	reg2	contador
0	Prod.	**lw** reg1, contador	5	?	5
1	Prod.	**addi** reg1, reg1, 1	6	?	5
2	Cons.	**lw** reg2, contador	?	5	5
3	Cons.	**addi** reg2, reg2, -1	?	4	5
4	Cons.	**sw** reg2, contador	?	4	4
5	Prod.	**sw** reg1, contador	6	?	6

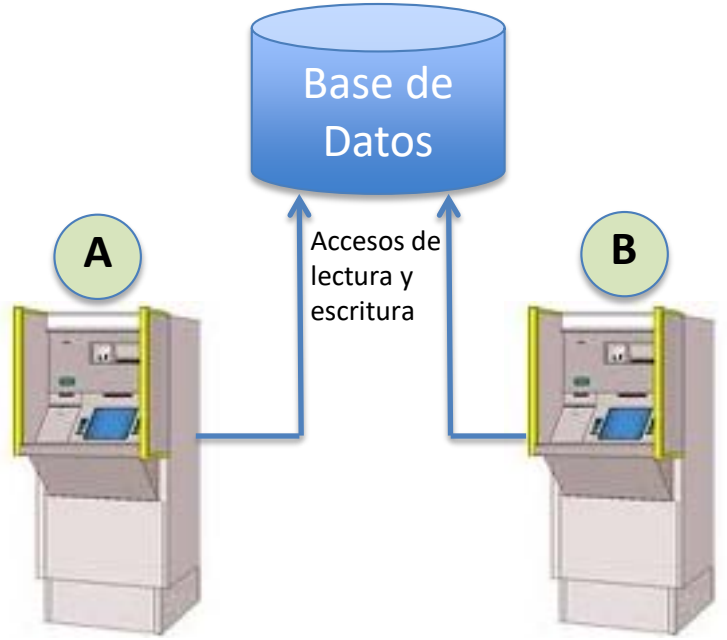
Incorrecto

Concepto de Condición de Carrera

- Supongamos que queremos sacar dinero de dos cajeros automáticos simultáneamente.

Sacar 20 Euros:

```
S=ConsultarSaldo()
Si S>=20 {
   DarDinero()
   NuevoSaldo(S-20)
}
```



- Supongamos que el saldo actual es 100 Euros y que se realizan las siguientes operaciones:

Cambios de contexto

A: ConsultarSaldo()
B: ConsultarSaldo()
B: DarDinero()
B: NuevoSaldo(80)
A: DarDinero()
A: NuevoSaldo(80)

¡Hemos sacado 40 euros pero todavía tenemos 80!
Obviamente los cajeros reales no se comportan así

Desde que un cajero consulta el saldo hasta que escribe el nuevo saldo, no deberíamos dejar que otro cajero iniciara una operación de sacar dinero

•Definición de Condición de Carrera

Una condición de carrera se produce cuando la ejecución de un conjunto de operaciones concurrentes sobre una **variable compartida** deja la variable en un estado que no se corresponde con el estado en el que quedaría después de alguna de las posibles ejecuciones secuenciales (**estado inconsistente**).

- El **problema de las condiciones de carrera** aparece porque
  - El programador se preocupa de la corrección secuencial de su programa, pero no sabe cuándo van a producirse los cambios de contexto
  - El sistema operativo no conoce las dependencias entre los procesos/hilos que está ejecutando, ni si es conveniente o no realizar un cambio de contexto en un momento determinado

Concepto de Condición de Carrera

- El error es muy **difícil de depurar**, porque el código de cada hilo es correcto por separado
  - La inconsistencia suele producirse muy de vez en cuando, porque sólo ocurre si hay un cambio de contexto en un lugar preciso (e inoportuno) del código.
  - Por lo tanto, el hecho de probar el código y que funciona bien, no asegura que esté libre de problemas de condición de carrera.
- Solución a la condición de carrera
  - No podemos, en general, controlar cuándo se producen cambios de contexto. Por tanto, tenemos que conseguir que los programas concurrentes sean correctos a pesar de que se produzcan cambios de contexto en cualquier lugar del código.

Es necesario sincronizar el acceso a variables compartidas

Ejercicio 1:Planificación con hilos

- A la cola de preparados de un sistema que soporta **hilos a nivel de núcleo** llegan 4 hilos H1, H2, H3 y H4, con las siguientes características:

Hilos	Instante Llegada	Ráfagas
H1	0 (1º)	6 CPU + 2 E/S + 1CPU
H2	0 (2º)	6 CPU + 2 E/S + 1CPU
H3	0 (3º)	2 CPU + 3 E/S + 1CPU + 3E/S + 1CPU
H4	0 (4º)	2 CPU + 3 E/S + 1CPU + 3E/S + 1CPU

El dispositivo de E/S es único y atiende las peticiones con un algoritmo FCFS. Indique cuál será **el tiempo promedio de espera** si el núcleo del sistema dispone de un planificador que utiliza uno de los siguientes algoritmos de planificación:

a) SRTF
b) RR (q=2)

Ejercicio 2: Planificación con hilos

- A la cola de preparados de un sistema que **NO soporta hilos a nivel de núcleo** llegan 4 hilos H1, H2, H3 y H4 con las siguientes características

Proceso	Hilos	Instante Llegada	Ráfagas
A	H1	0 (1º)	6 CPU + 2 E/S + 1CPU
A	H2	0 (2º)	6 CPU + 2 E/S + 1CPU
B	H3	0 (3º)	2 CPU + 3 E/S + 1CPU + 3E/S + 1CPU
B	H4	0 (4º)	2 CPU + 3 E/S + 1CPU + 3E/S + 1CPU

El ***runtime*** del lenguaje de programación tiene un **planificador FCFS**. El dispositivo de E/S es único y atiende las peticiones con un algoritmo FCFS.

Indique cuál será **el tiempo promedio de espera** si el núcleo del sistema utiliza uno de los siguientes algoritmos de planificación:

a) SRTF

b) RR (q=2)