Fonaments dels Sistemes Operatius (FSO)

Departament d'Informàtica de Sistemes i Computadores (DISCA)
Universitat Politècnica de València

Bloc Temàtic 2: Processos

Unitat Temàtica 5

Fils d'execució

fSO

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DISCA

- **Objectius**
  - Introduir el concepte de **programació concurrent**
  - Introduir el concepte de **fil d'execució** i els seus diferents models d'implementació
  - Estudiar les diferències entre fil d'execució i procés pesat
  - Examinar la problemática asociada al **us compartit de memòria per** part d'**activitats concurrents**

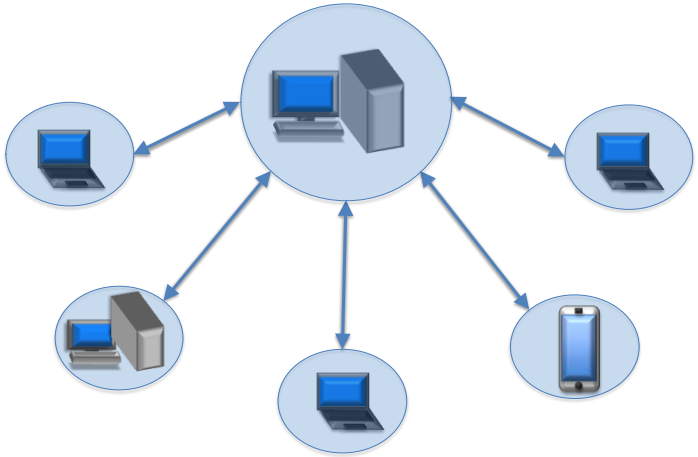
Bibliografía i contenido

- Contenido
  - Programació concurrent
  - Concepte de *fil d'execució*
    - Procés vs. Fil d'execució
  - Models de fils d'execució
  - Necessitat de sincronització
  - Concepte de *condició de carrera*

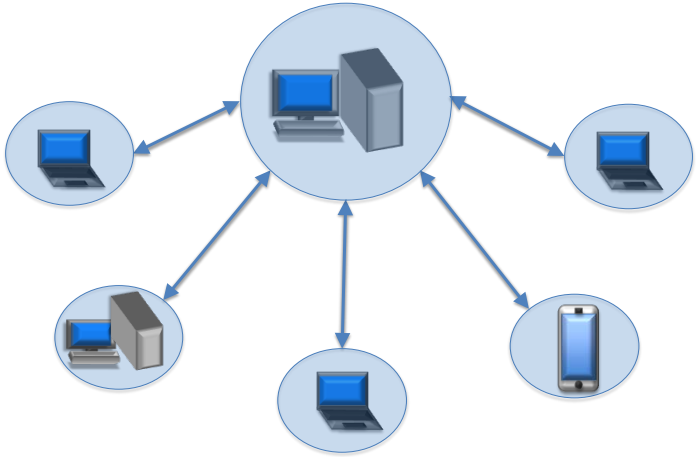
- Bibliografia
  - "Fundamentos de sistemas operativos" Silberschatz 7ª Ed
  - "Sistemas operativos: una visión aplicada" Carretero 2º Ed

- **Programació concurrent**
- Concepte de *fil d'execució*
- Models de fils d'execució
- Necessitat de sincronització
- Concepte de *condició de carrera*

- **Programació concurrent:**
  - Un **únic programa** que intenta resoldre un problema definint **diverses "activitats"**
  - Existeix un **paral·lelisme** potencial **entre tasques**
- **Exemples d'aplicacions** :
  - **Servidor web** capaç d'atendre mes d'una petició de client, de manera que cada petició es atesa per una *activitat*
  - Jocs d'ordinador/consola en els que cada personatge/objecte representa una *activitat*



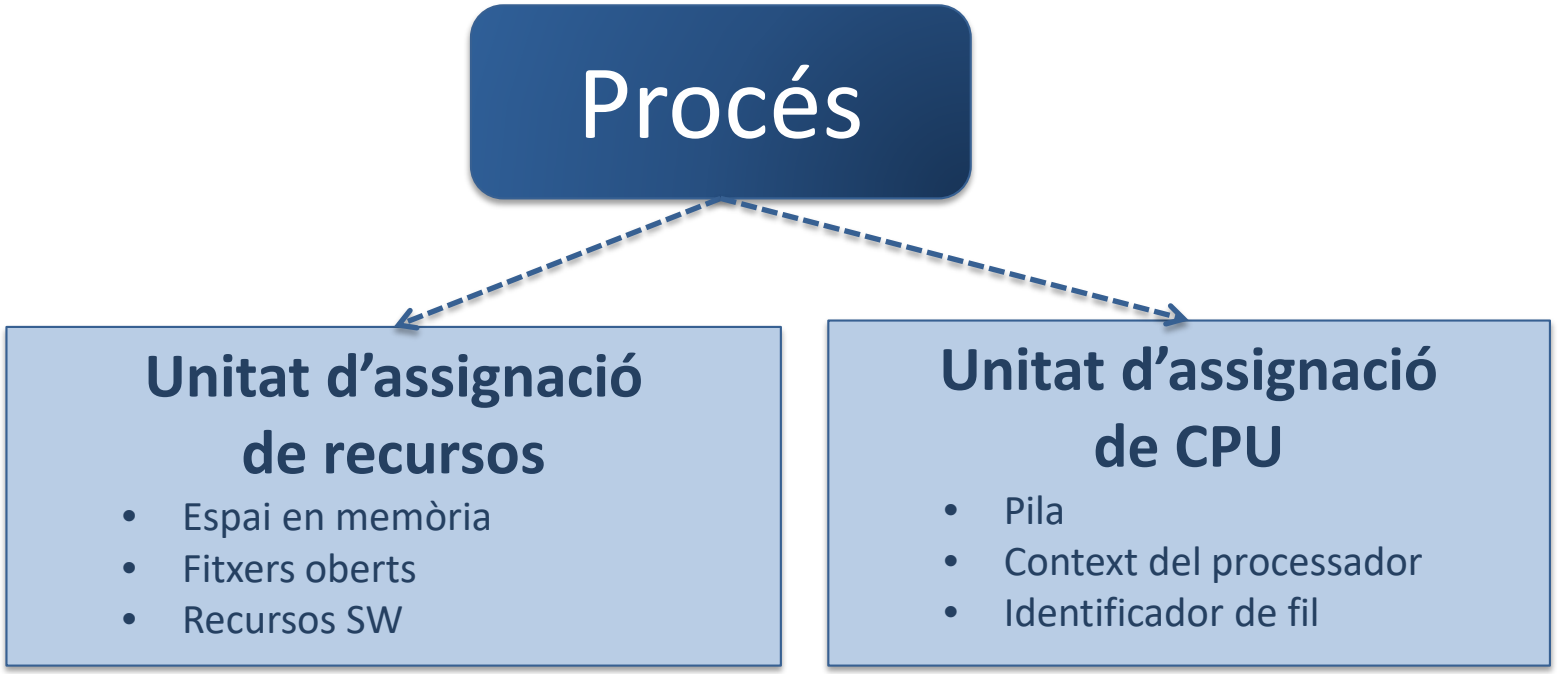
- Les “activitats” d’un programa concurrent
  - treballen en comú
    - » i requereixen **comunicar-se** entre elles per a intercanviar dades (a través de memòria compartida i/o pas de missatges.)
    - » **sincronitzar** les seues línies de fluix de control**.**
- Per a implementar aquestes activitats duess possibilitats
  - Activitat = **Procés**
  - Activitat = **Fil d'execució (“thread”)**



- Programació concurrent
- **Concepte de *fil d'execució***
- Models de fils d'execució
- Necessitat de sincronització
- Concepte de *condició de carrera*

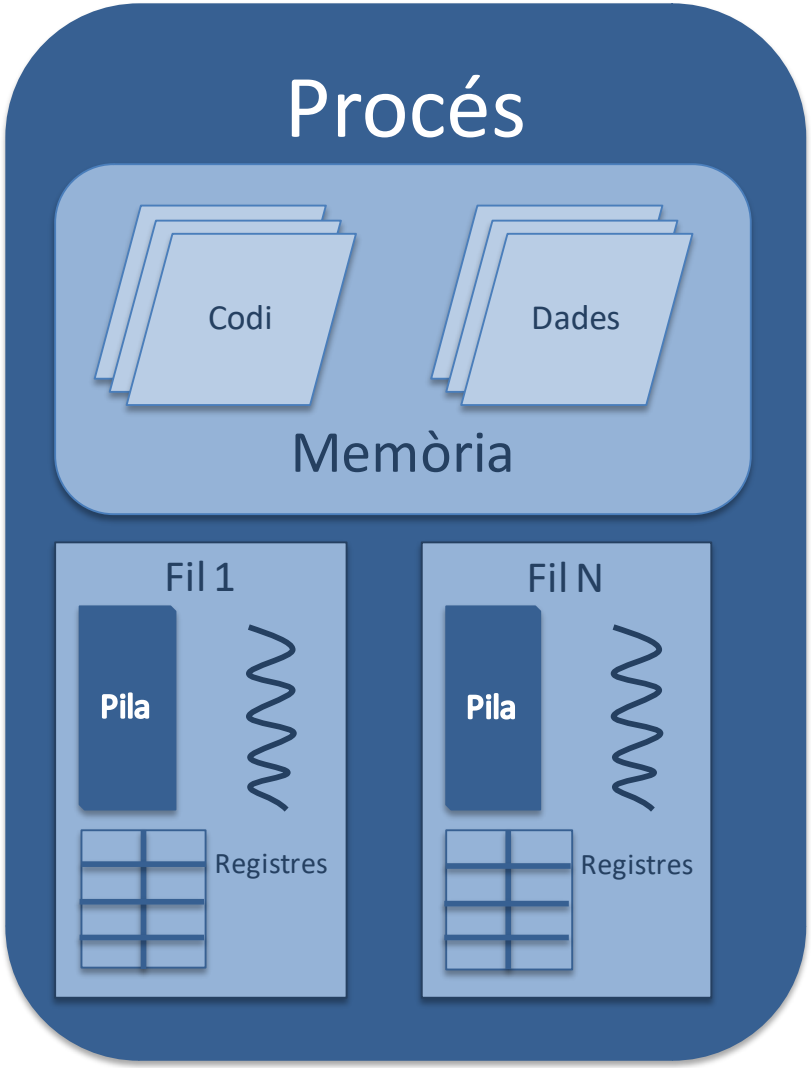
Concepte de fil d'execució

- Un procés es una **entitat d'abstracció** composta per:



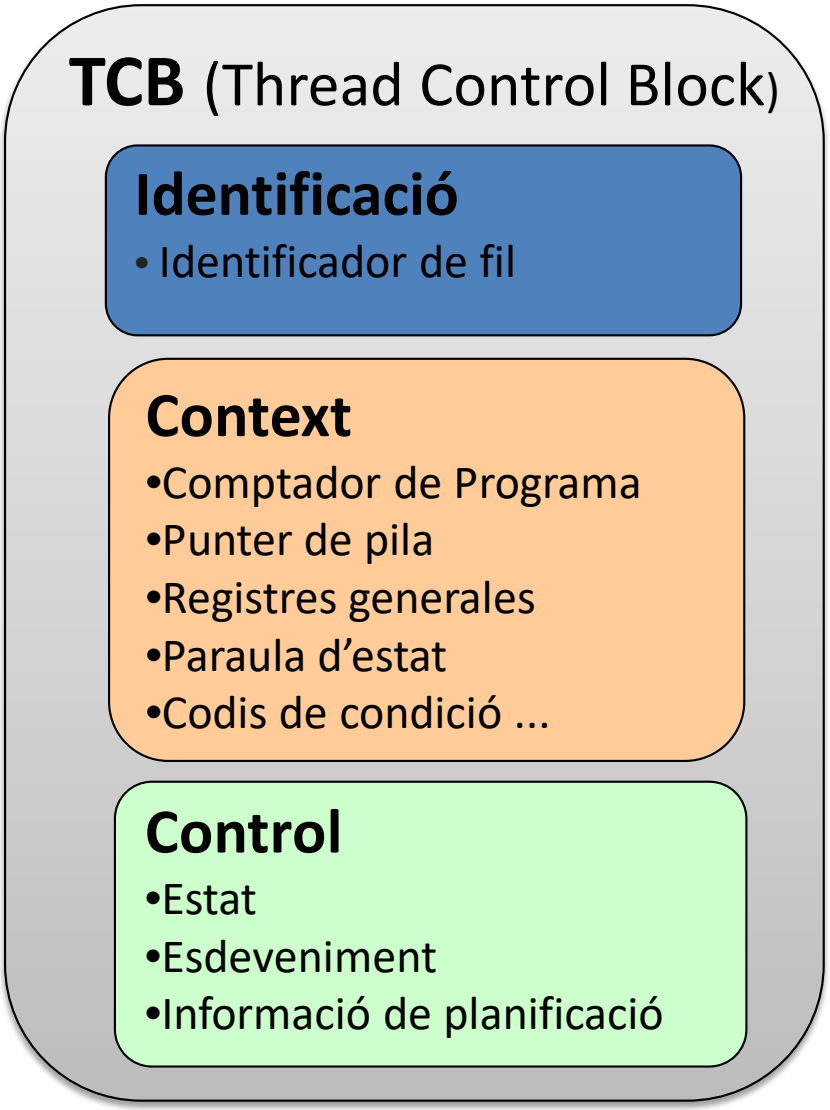
- El sistema operatiu pot disociar aquestes dues unitats d'assignació del procés

Concepte de fil d'execució



- **Fil d'execució:** Unitat bàsica d'assignació de CPU.
- **Procés** = Unitat d'assignació de recursos amb al menys un fil d'execució
- Els fils d'execució definits **dins d'un mateix procés** comparteixen:
  - Codi
  - Dades,
  - Recursos assignats al procés
- Cada fil disposa d'**atributs propis**:
  - Identificador (ID)
  - Pila
  - Comptador de programa
  - Registres

- Implementació de fils d'execució



- **Atributs**
  - Los fils tenen **pocs atributs**
  - La informació necessària per a suportar fils d'execució és mes reduida que la que s'ha de mantenir per als processos pesats
  - La informació dels recursos compartits es guarda en el PCB del procés

- **Procés versus fil**
  - Des del **punt de vista del sistema**, és **més barat**
    - Costa menys...
      - » **crear un fil** en un procés existent que crear un procés nou
      - » **terminar un fil** que un procés.
      - » canviar de context entre dos fils d'un mateix procés que entre dos processos
  - Des del **punt de vista del programador**, és més natural
    - Els fils presenten un model de programació concurrent mes senzill, en el qual la comunicació és:
      - » Més natural (els fils comparteixen memòria/fitxers per definició)
      - » Més eficient (en moltes ocasions, no fa falta sol·licitar serveis al nucli)

Contingut

- Programació concurrent
- Concepte de *fil d'execució*
- **Models de fils d'execució**
- Necessitat de sincronització
- Concepte de *condició de carrera*

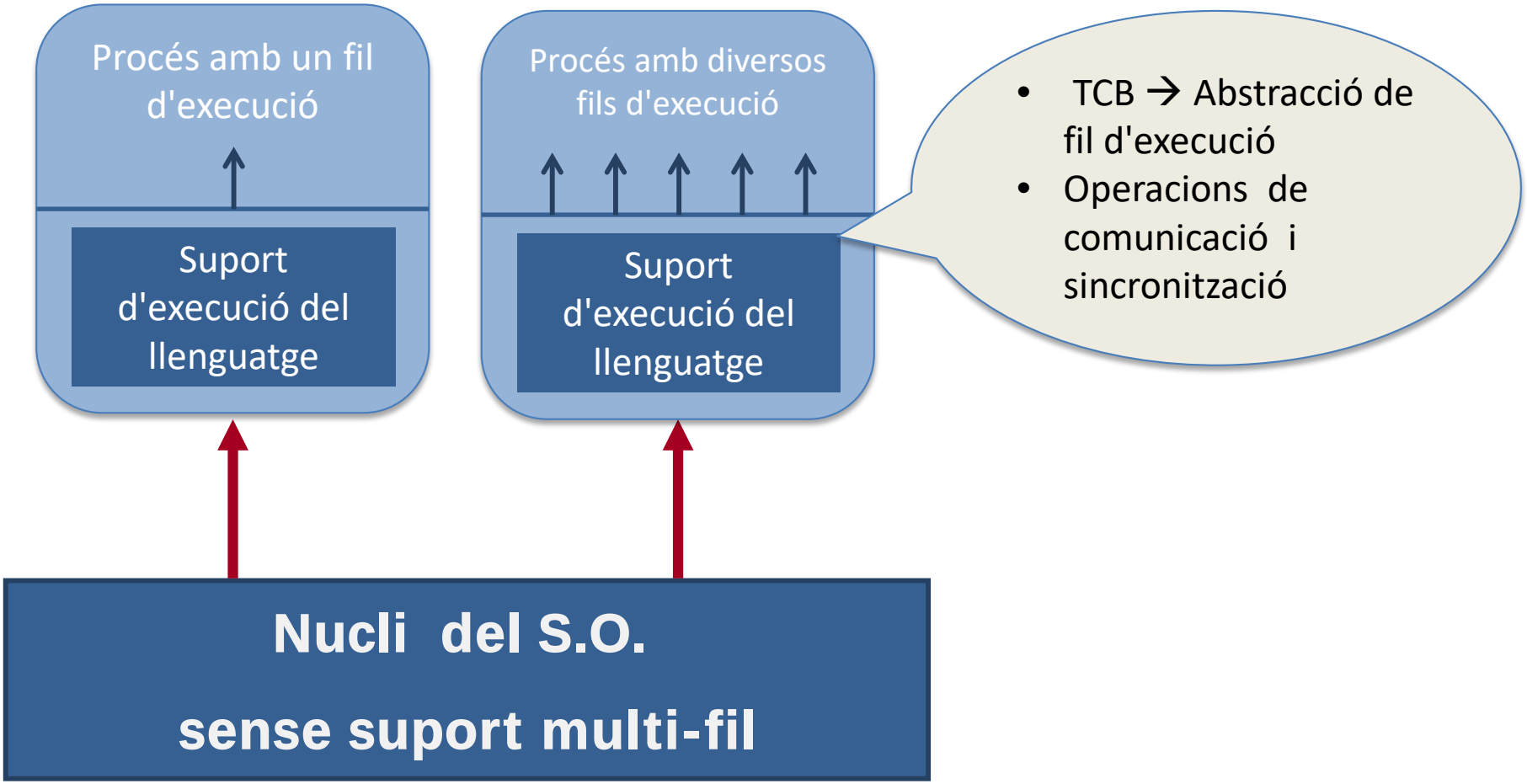
Models de fils d'execució

- Per a programar amb fils és necessari **suportar** l'abstracció de **fil d'execució**, en base a:
  - Estructures de dades amb atributs dels fils (TCBs)
  - Operacions de comunicació i de sincronització de fils

- **Tres models d'implementació:** En funció de qui ofereix/supor-te aquestes abstraccions
  - Fils a **nivell de usuari**
  - Fils a **nivell de nucli**
  - Fils **híbrids**

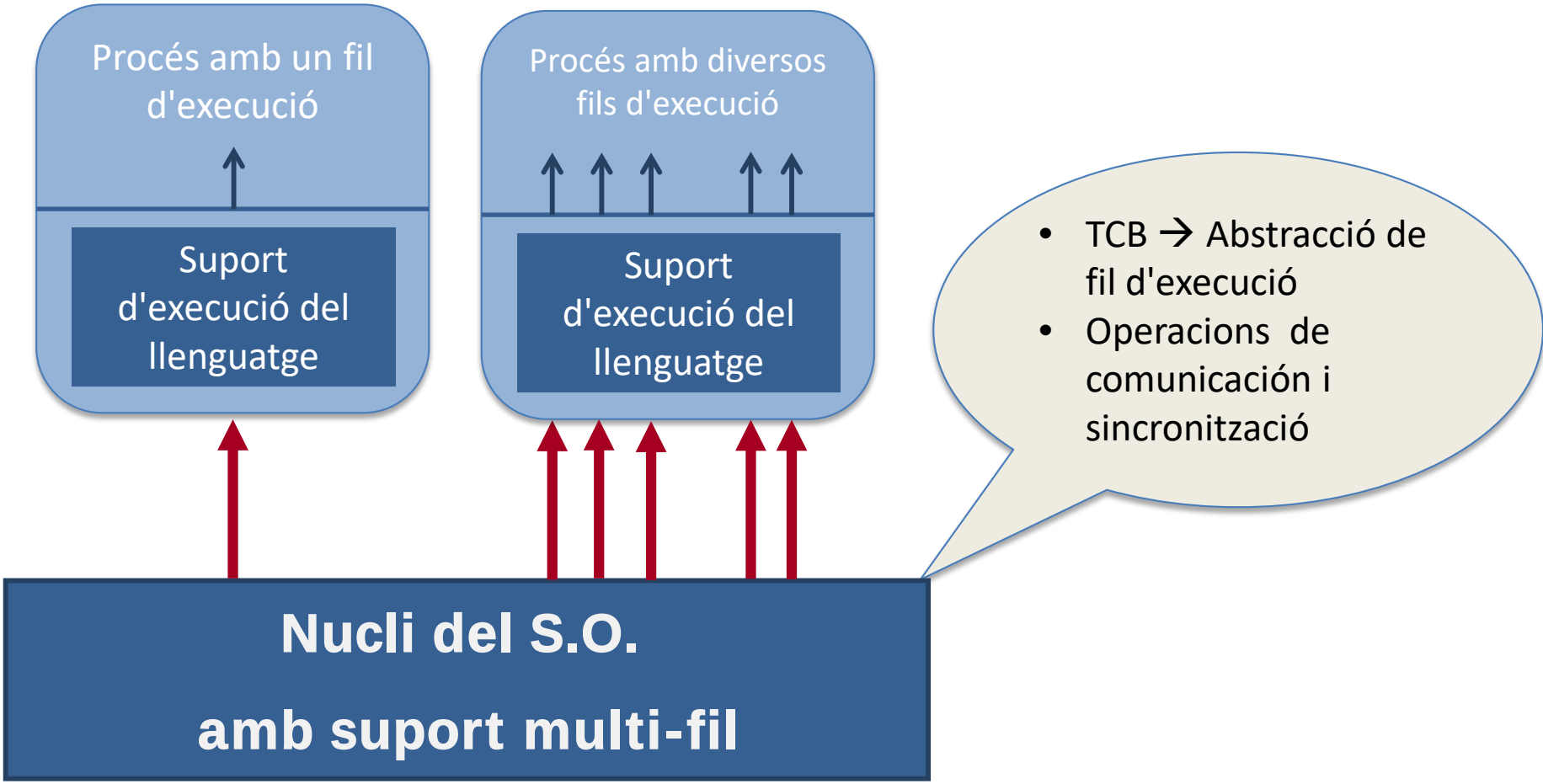
Models de fils d'execució

- **Fils a nivell d'usuari**
  - Les abstraccions les ofereix el suport d'execució del **llenguatge de programació** ("run time")

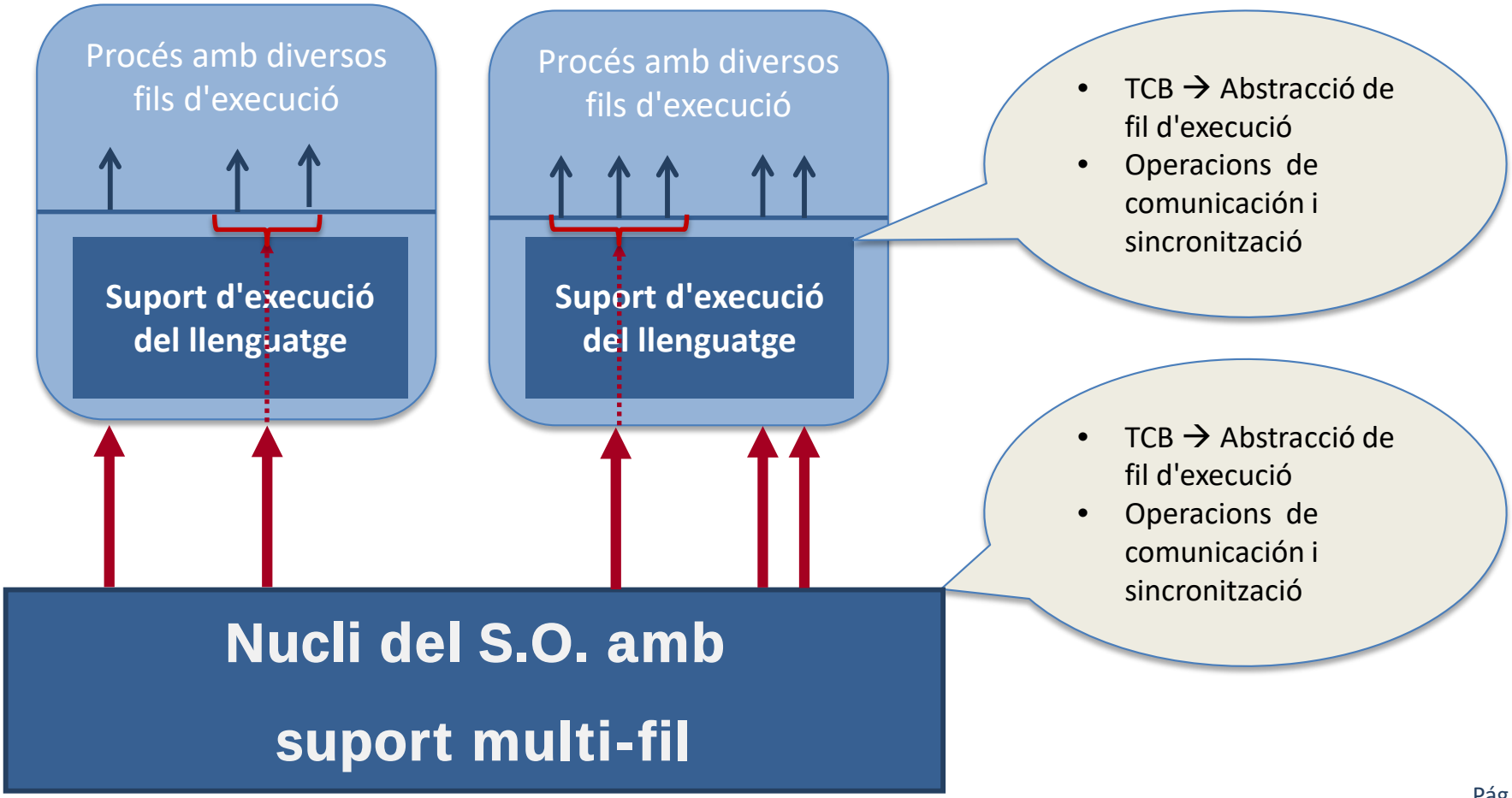


Models de fils d'execució

- **Fils a nivell de nucli**
  - Les abstraccions les ofereix el **nucli** del sistema operatiu mitjançant la interfície de crides al sistema



- **Model híbrid**
  - Les abstraccions les ofereixen el nucli del sistema operatiu i el suport d'ejecució del **llenguatge de programació** ("run time")



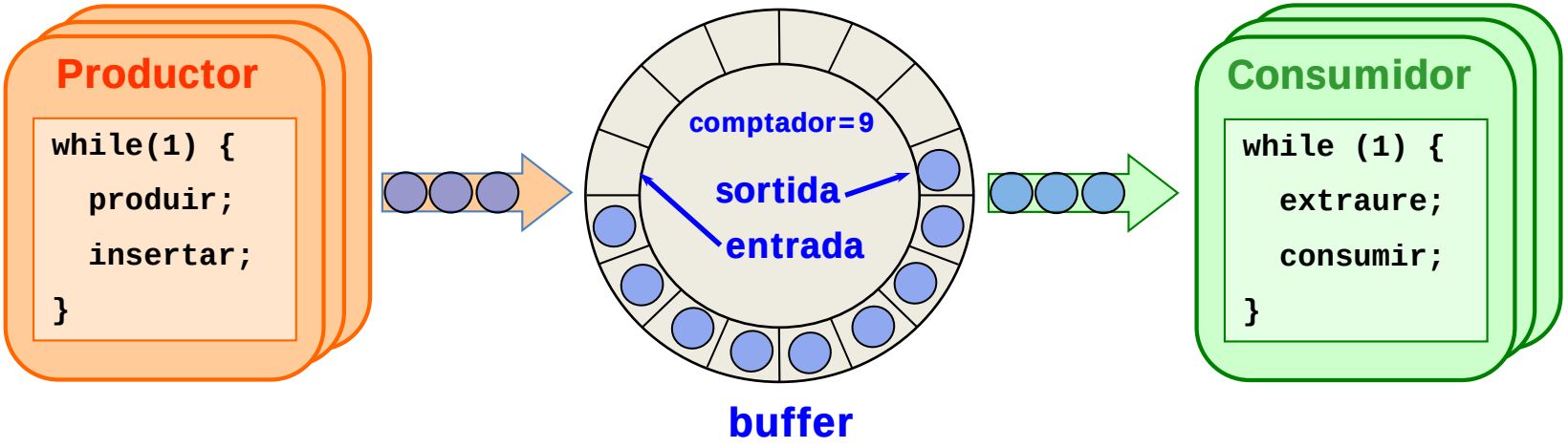
- Programació concurrent
- Concepte de *fil d'execució*
- Models de fils d'execució
- **Necessitat de sincronització**
- Concepte de *condició de carrera*

Necessitat de sincronització

- La **concurrència es fonamental**
  - Tant per a les aplicacions d'usuari com en l'estructura interna del Sistema Operatiu se utiliza concurrència.
- La **programació concurrent aborda** els següents aspectes:
  - Comunicació entre processos
  - Compartició de recursos
  - Sincronització d'activitats
  - Reserva de temps de CPU
- La concurrència **es manifesta tant**
  - en **entorns de multiprocessadores** o distribuits
  - com en **entorns monoprocessadores** i de temps compartit
- Es poden distinguir tres contextes de concurrència:
  - Múltiples aplicacions
  - Una sola aplicació que s'estructura en diverses activitats (diversos fils o diversos processos)
  - Estructura del Sistema Operatiu. El SO està implementat en la forma de diverses activitats

Necessitat de sincronització

- Exemple: problema del **"productor/consumidor"** ("buffer acotat")
  - Existeixen dos tipus d'entitats: productors i consumidors (de "items")
  - Existeix un buffer acotat (circular) que acomoda la diferència de velocitat entre productors i consumidors:
    - Si el buffer es plena, els productors han de suspendre's
    - Si el buffer es buida, els consumidors han de suspendre's



Necessitat de sincronització

- Codi dels fils productor i consumidor

Comparteixen els fils productors i consumidors

```
#define N  20
int  buffer[N];
int  entrada, sortida, comptador = 0;
```

Bucles "d'**espera activa**"

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

  while(1) {
    item = producir();

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

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

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

    consumir(item);
  }
}
```

En aquest codi:

"`comptador`" i "`buffer`" son compartits pel fil productor i consumidor

Amb diversos fils productors i consumidors, "`entrada`" sería compartida per tots els productors, i "`sortida`" per tots els consumidors

Necessitat de sincronització

- **Productor/Consumidor,**
  - Fils productors i consumidors s'executen de forma **concurrent**
    - accedint a variables compartides
  - Els fils són triats per a la seua **execució independentment**
  - Les decisions de **quin fil s'executa** en cada moment, **i quan,** es prenen en cada **canvi de context. Depenen d'un planificador** i no del programador de l'aplicació.

Una **condició de carrera** es dóna quan *"Un codi que és correcte si s'executa de forma seqüencial, pot deixar de ser-ho quan s'executa concurrentment"*

- Programació concurrent
- Concepte de *fil d'execució*
- Models de fils d'execució
- Necessitat de sincronització
- **Concepte de *condició de carrera***

Concepte de condició de carrera

Si suposem que:

- ✓ Inicialment "comptador" val 5
- ✓ Un productor executa "comptador=comptador + 1;"
- ✓ Un consumidor executa "comptador=comptador - 1;"

el resultat final del "comptador" deuria ser 5

Productor:

```
comptador = comptador + 1;
```

```
lw reg1, comptador
addi reg1, reg1, 1
sw reg1, comptador
```

Consumidor:

```
comptador = comptador - 1;
```

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

Però si s'executa la següent seqüència d'operacions...

T	Fil	Operació	reg1	reg2	comptador
0	Prod.	**lw** reg1, comptador	5	?	5
1	Prod.	**addi** reg1, reg1, 1	6	?	5
2	Cons.	**lw** reg2, comptador	?	5	5
3	Cons.	**addi** reg2, reg2, -1	?	4	5
4	Cons.	**sw** reg2, comptador	?	4	4
5	Prod.	**sw** reg1, comptador	6	?	6

Canvis de context

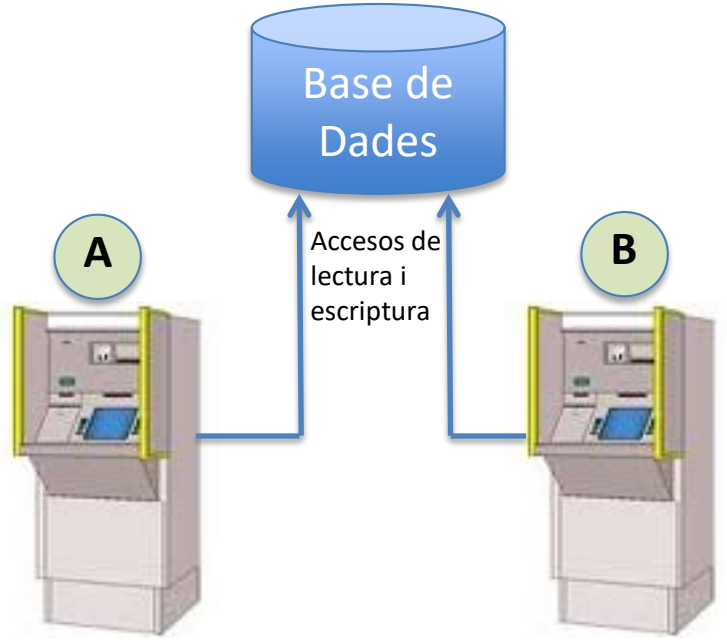
Incorrecte

Concepte de condició de carrera

- Suposem que volem traure diners de dos caixers automàtics simultàniament.

Traure 20 Euros:

```
S=ConsultarSaldo()
Si S>=20 {
   DonarDiners()
   NouSaldo(S-20)
}
```



- Suposem que el saldo actual és 100 Euros i que es realitzen les següents operacions:

Canvis de context

A: ConsultarSaldo()
B: ConsultarSaldo()
B: DonarDiners()
B: NouSaldo(80)
A: DonarDiners()
A: NouSaldo(80)

¡Hem tret 40 euros però encara tenim 80!
Obviament els caixers reals no es comportan així

Des de que un caixer consulta el saldo fins que escriu el nou saldo, no deuriem deixar que un altre caixer iniciara una operació de traure diners

Concepte de condició de carrera

•Definició de *condició de carrera*

Una condició de carrera es produeix quan l'execució d'un conjunt d'operacions concurrents sobre una **variable compartida** deixa la variable en un estat que no es correspon amb l'estat en el que quedaria després d'alguna de les possibles execucions seqüencials (**estat inconsistent**).

- El **problema de les condicions de carrera** apareix, perque
  - El programador es preocupa de la correcció seqüencial del seu programa, però no sap quan van a produir-se els canvis de context
  - El sistema operatiu no coneix les dependències entre els processos/fils que està executant, ni si és convenient o no realitzar un canvi de context en un moment determinat

Concepte de condició de carrera

- L'error es molt **difícil de depurar**, perque el codi de cada fil és correcte per separat
  - La inconsistència sol produir-se molt de tant en tant, perque només ocórre si hi ha un canvi de context en un lloc precís (e inoportú) del codi.
  - Per tant, el fet de provar el codi i que funcione be, no assegura que estiga lliure de problemes de condició de carrera.
- Solució a la condició de carrera
  - No podem, en general, controlar quan es produeixen canvis de context. Per tant, hem d'aconseguir que els programas concurrents siguen correctes malgrat que es produisquen canvis de context en qualsevollloc del codi.

És necessari sincronitzar l'accés a variables compartides

Ejercici 1:Planificació amb fils

- A la cua de preparats d'un sistema que suporta **fils a nivell de nucli** arriben 4 fils H1, H2, H3 y H4, amb les següents característiques:

Fils	Instant Arribada	Ràfegues
H1	0 (1er)	6 CPU + 2 E/S + 1CPU
H2	0 (2on)	6 CPU + 2 E/S + 1CPU
H3	0 (3er)	2 CPU+3 E/S+1CPU+3E/S +1CPU
H4	0 (4t)	2 CPU+3 E/S+1CPU+3E/S +1CPU

El dispositiu d'E/S és únic i atén les peticions amb un algorisme FCFS.
Indiqueu quin serà **el temps promig d'espera** si el nucli del sistema disposa d'un planificador que utilitza un dels següents algorismes de planificació:

a) SRTF
b) RR (q=2)

Exercici 2: Planificació amb fils

- A la cua de preparats d'un sistema que **NO suporta fils a nivel de nucli** arriben 4 fils H1, H2, H3 y H4, amb les següents característiques:

Procés	Fils	Instant Arribada	Ràfegues
A	H1	0 (1er)	6 CPU + 2 E/S + 1CPU
A	H2	0 (2on)	6 CPU + 2 E/S + 1CPU
B	H3	0 (er)	2 CPU+3 E/S+1CPU+3E/S +1CPU
B	H4	0 (4t)	2 CPU+3 E/S+1CPU+3E/S +1CPU

El **run-time** del llenguatge de programació te un **planificador FCFS**. El dispositiu d'E/S és únic i atén les peticions amb un algorisme FCFS.

Indiqueu quin serà **el temps mitjà d'espera** si el nucli del sistema utilitza un dels següents algorismes de planificació:

a) SRTF

b) RR (q=2)