Programació concurrent i a temps real

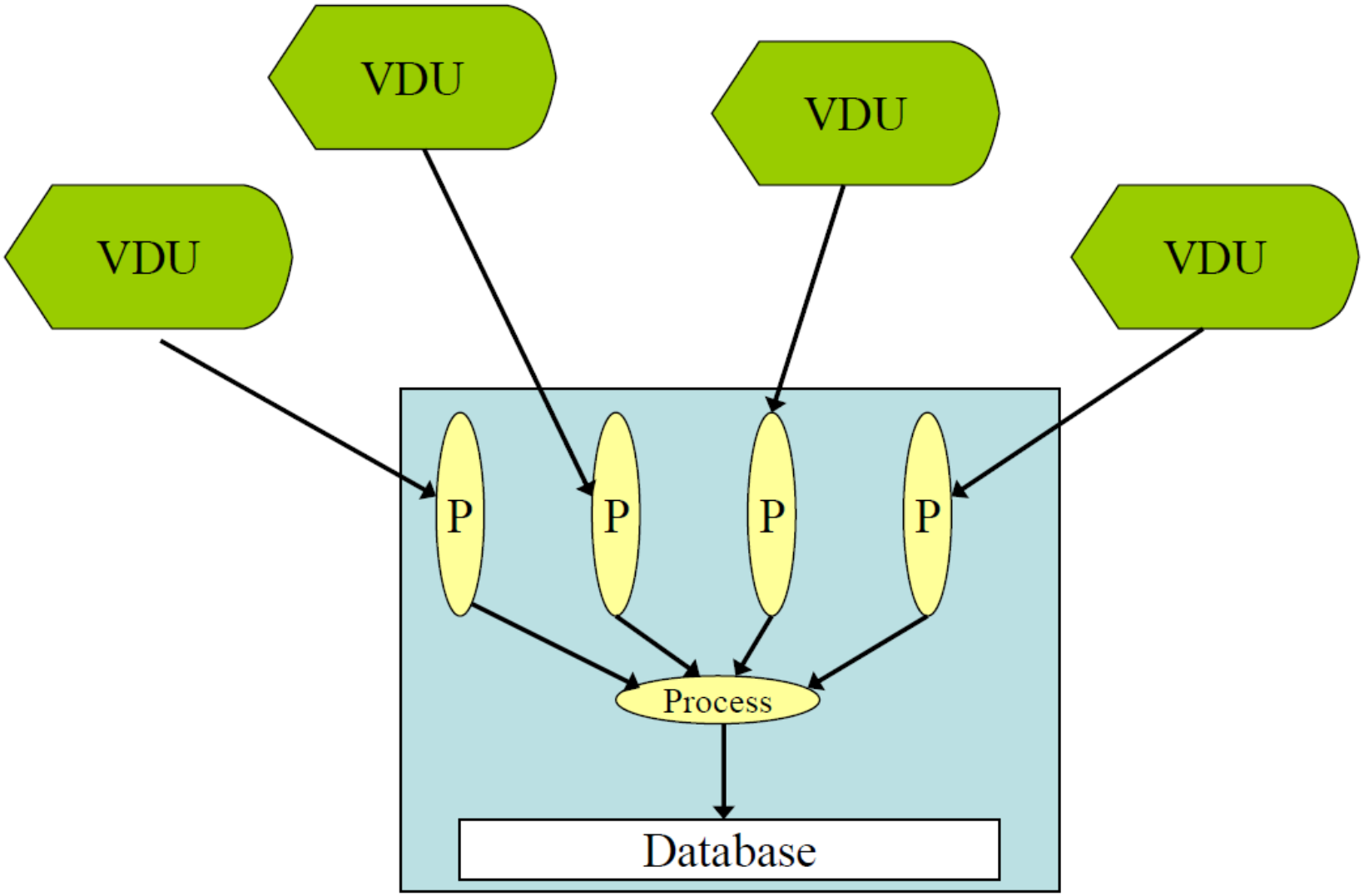
Processos:

Estats i operacions

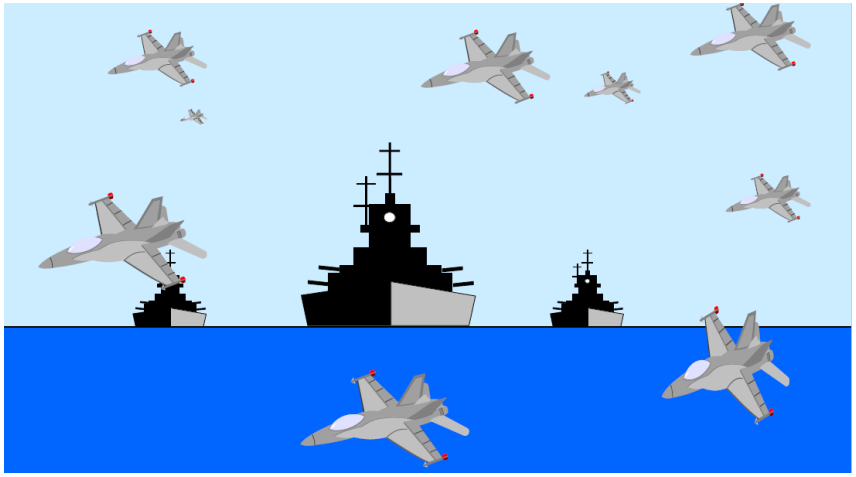
Programació concurrent

- La programació concurrent recull un conjunt de tècniques informàtiques usades per representar i gestionar el paral·lelisme i les eines de sincronització i comunicació entre programes.
- L'ús de la programació concurrent simplifica la implementació d'alguns tipus de comportaments.
- Permet usar paral·lelisme en els casos que es necessita.
- El mon real actua en paral·lel.
- Tots els sistemes de temps real son inherentment concurrents, els dispositius actuen en paral·lel en el mon real.
- Sembla lògic, que els nostres sistemes també actuïn en paral·lel.

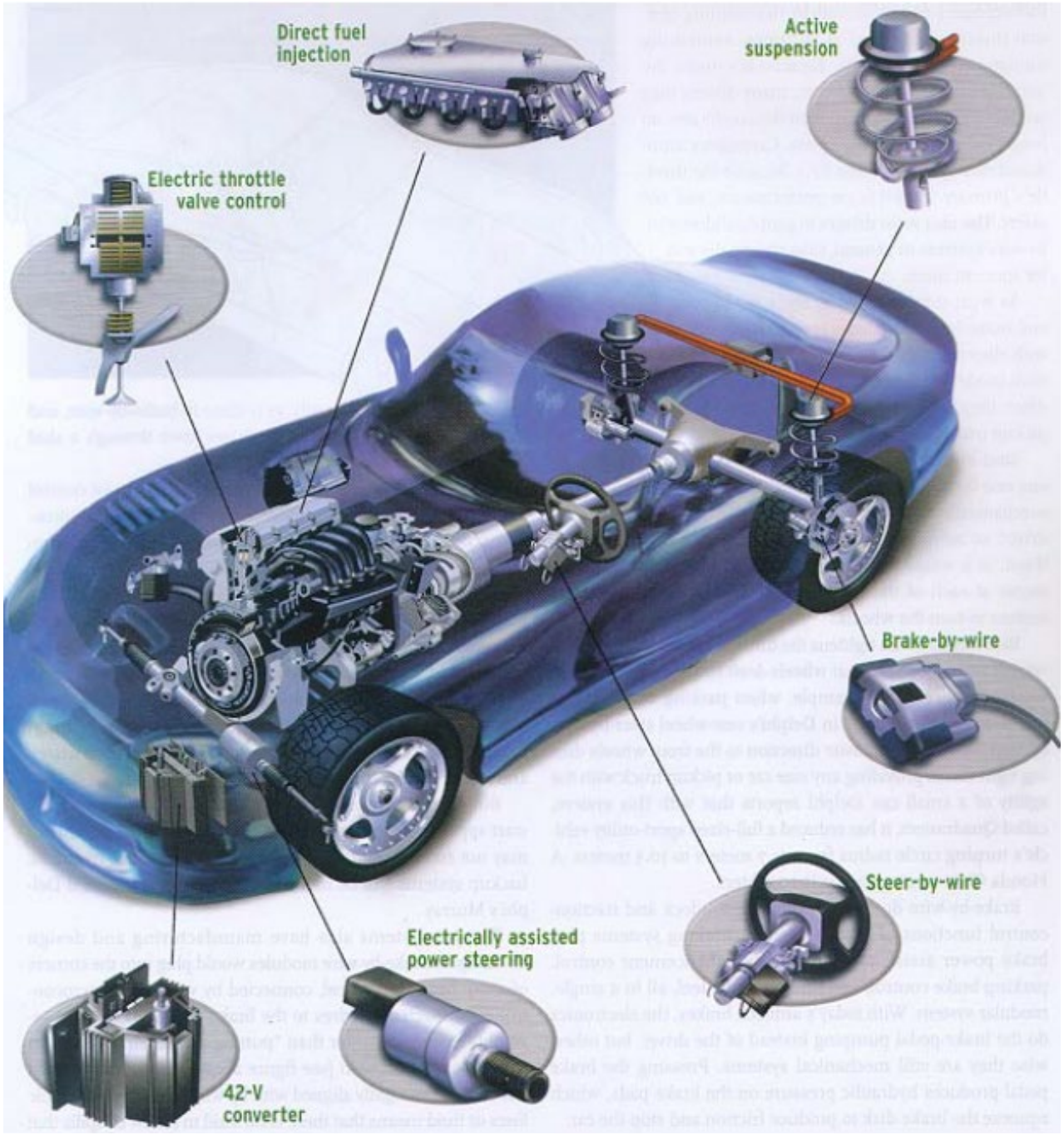
Sistema de reserves aèries



Sistema de Control de tràfic



Sistemes de control



Processos i concurrència

Terminologia

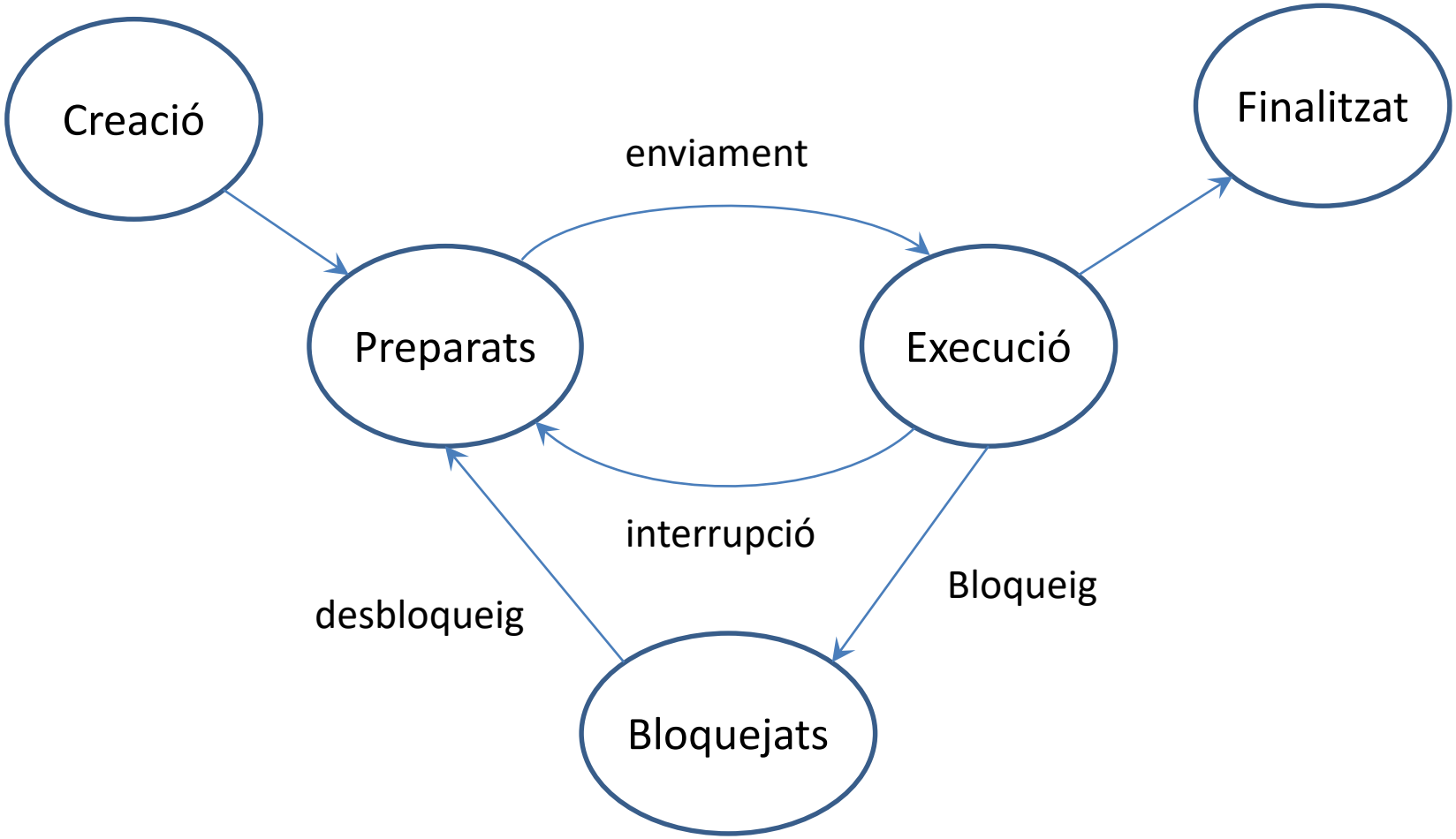
- Un procés és un programa en execució.
- Un sistema concurrent és una col·lecció de processos seqüencials autònoms en execució simultània (paral·lel o no)
- Cada procés té un únic fil d'execució seqüencial (*thread)*
- La implementació d'aquest sistema pot ésser :
  - El processos multiplexen la seva execució en un únic processador (*Multiprogramming)*
  - Els processos multiplexen la seva execució entre diferents processadors que comparteixen la memòria i altres recursos (*Multiprocessing)*
  - Els processos multiplexen la seva execució entre diferents processadors que no comparteixen la memòria (*Distributed Processing)*

Processos

- Parts d'un procés:
  - Imatge de memòria
  - Taules de control

- Imatge de memòria
  - Memòria física ocupada pel codi i les dades del programa
  - Hi ha 4 parts:
    - Codi o text: Programa carregat a memòria
    - Dades: Constants i variables estàtiques del programa
    - Pila: Paràmetres i resultats de les funcions i variables locals
    - Zona de memòria dinàmica
- Taules de control
  - Estructures variades i complexes amb la informació d'estat de cada procés:
    - Valors dels registres del processador
    - Taules de pàgines de memòria i d'E/S
    - Informació de la propietat del procés
    - Estadístiques d'us
    - ....

Estats d'un procés



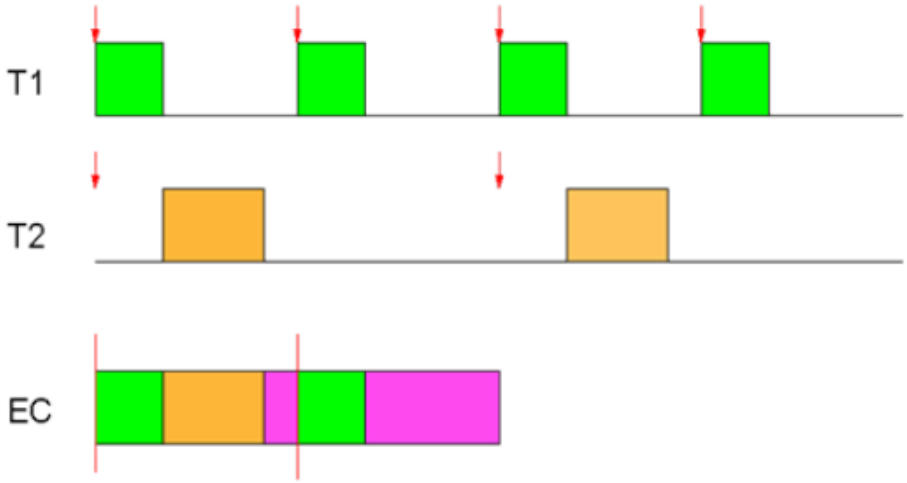
Estats d'un procés

Durant el seu temps de vida, un procés pot estar en un dels diferents estats possibles: Elaboració, execució, preparat, bloquejat, completat i finalitzat

- Un procés està en activació o elaboració quant s'està elaborant la seva part declarativa
- Un procés s'està en execució si les seves sentències s'estan executant en un processador
- Un procés està preparat si s'està esperant que se li assigni un processador
- Un procés està bloquejat si s'està esperant que succeeixi algun esdeveniment, com una sincronització amb un altre procés. També es pot dir que està adormit
- Un procés s'ha completat quan si s'ha acabat d'executar la seva seqüència de sentències, però no pot acabar perquè està actiu algun del seu processos dependents
- Un procés ha finalitzat si ja no està actiu.

Els processos porten associades diverses operacions: Creació i activació, finalització, sincronització, comunicació i planificació

Programació concurrent (Alternatives)



Mentre noFinal fer

Cas cicle

0: T1, T2, E0

1: T1, E1

Fcas

Cicle ++

fMentre

- Programació seqüencial
- Executiu cíclic per gestionar les diferents activitats concurrents
- Complica la tasca del programador
- Programa difícil de llegir i analitzar
- No hi haurà execució en paral·lel
- Implementar codi tolerant a fallades serà molt complexa

Planificador

- La transició de preparats a execució la decideix el planificador (Scheduler).
- S'executa:
  - Pel bloqueig d'un procés
  - Per un esdeveniment (normalment una interrupció del maquinari generada per un dispositiu)
  - Per un desbloqueig
- Per evitar que un procés retingui el processador indefinidament -> Es generen interrupcions periòdiques amb un rellotge.

Planificador

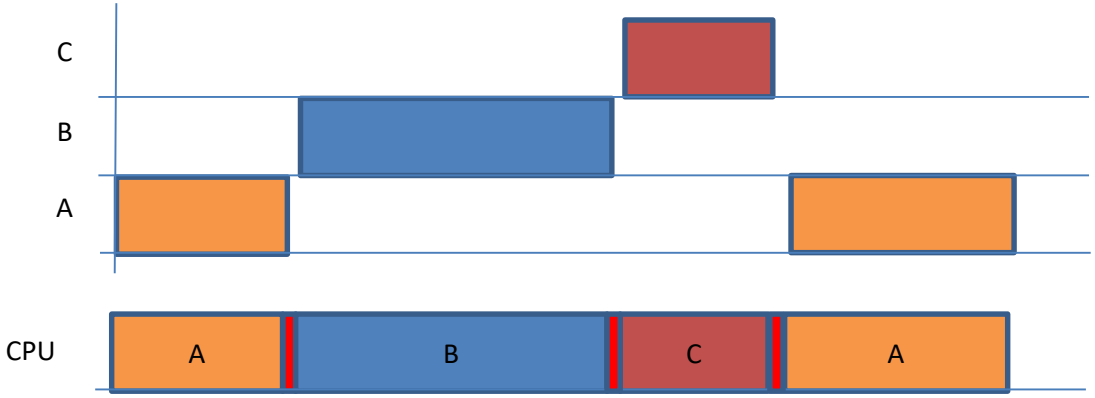
- Interrupció -> Cada pocs mil·lisegons
- El planificador ha de decidir quin procés passa a execució
- Planificador apropiatiu -> Treu un procés que s'està executant i n'hi posa un altre de la llista de preparats
- El planificador ha de prendre moltes (entre pocs centenars i desenes de milers) decisions per segon. Han de tenir una complexitat computacional baixa. O(1).
- El rendiment i la velocitat del sistema depèn de les decisions del planificador
- Els requisits:
  - Temps de resposta
  - Equitat
  - Diferents prioritats
  - Evitar les esperes infinites

Planificador: canvi de context

- És el procediment d'emmagatzemar l'estat del procés que està en execució i restaurar el del procés que ha seleccionat el planificador
- Ha de fer:
  - Guardar els registres del processador
  - Marcar com invàlides les entrades de la memòria cau de les taules de pàgines (TLB) i copia les entrades modificades
  - Restaurar els registres i taules de pàgina del nou procés
- Genera:
  - Augment de la taxa de fallades de memòria cau i TLB

Intercalació

- Execució seqüencial en un processador



- Intercalació exclusiva
- El planificador selecciona el procés que s'executarà, aquest ho farà durant un període de temps denominat ràfega de CPU (CPU burst)
- Les combinacions d'intercalació entre els diferents processos és no determinista

Intercalació

- Execució multi procés

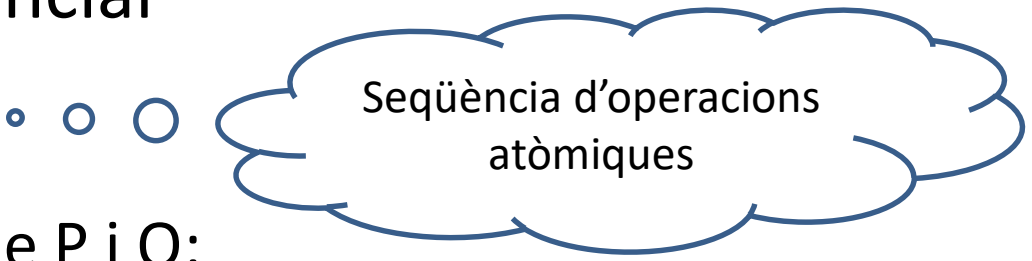
- Superposició d'execucions
- La superposició no complica la resolució dels problemes de sincronització i concurrència
- la intercalació i l'execució no determinista són l'origen real dels seus riscos

Problemes de la intercalació

- Consistència seqüencial

  P = { p0, p1, p2 }

  Q = { q0, q1, q2 }



  Una execució vàlida de P i Q:

  p0, p1, p2, q0, q1, q2 o q0, q1, q2, p0, p1, p2

  Consistència seqüencial:
  p1 s'ha d'executar després de p0 i abans de p2

  p0, q0, p1, q1, p2, q2; També ho compleix

- Si aquestes instruccions accedeixen o modifiquen variables compartides els resultats poden ser diferents -> Seqüència no determinista d'execució

Problemes de la intercalació

- Programa seqüencial

```
a = a + 1
b = b + a
print “a, b:”, a, b
```

Molts llenguatges de programació estan dissenyats per especificar i executar instruccions seqüencials

El resultat: a, b: [1 1], [2 3], [3 6], [4 10], [5 15], ...

- Si executem aquest codi en dos processos independents (P i Q) sobre un sistema mono processador i {a i b} son variables compartides:

		Execució		
Procés P	Procés Q	a	b	Mostra
....		0	0	
a = a + 1		1	0	
	a = a + 1	2	0	
	b = b + a	2	2	
	print "a, b:", a, b			2 2
	....			
b = b + a		2	4	
print "a, b:", a, b				2 4

Condició de carrera

Problemes de la intercalació

- Programa molt simple

  comp += 1

assemblador de processadors MIPS

```
lw r1, comp(r0)    ; Carrega el contingut de “comp” a r1
add r1, r1, 1      ; Incrementa el contingut del registre r1
sw r1, comp(r0)    ; Guarda el valor de r1 a “comp”
```

- Si s’executa dos cops: -> comp = 2

		r1	comp
lw r1, comp(r0)		0	0
	lw r1, comp(r0)	0	0
	add r1, r1, 1	1	0
	sw r1, comp(r0)	1	1
		0	1
add r1, r1, 1		1	1
sw r1, comp(r0)		1	1

Condició de carrera

Problemes de la intercalació

Exemples en diferents llenguatges

```
for i in range(5000000):
    comp += 1
```

- Si aquest codi s'executa amb dos fils diferents, al final de l'execució el valor de ***comp*** hauria de ser 10.000.000, però cap obté el valor correcte.
- El resultat de qualsevol dels seus execucions és similar a les següents:
  - En C: valor de ***comp***: 5.785.161
  - En Payton: valor de ***comp***: 7.737.989
  - Sobre una raspberry Pi i Payton: valor de ***comp***: 7.496.983

Planificació

- Els processos poden ésser :
  - Independents
  - Cooperatius

- En qualsevol moment es pot produir un canvi de procés.
- Durant el disseny dels processos no s'ha de suposar res sobre l'ordre relatiu en que s'executen.
- En els casos en que sigui necessari un ordre relatiu s'ha de forçar.
- En el cas de processos independents és dissenya com si el sistema fos l'únic procés del sistema.

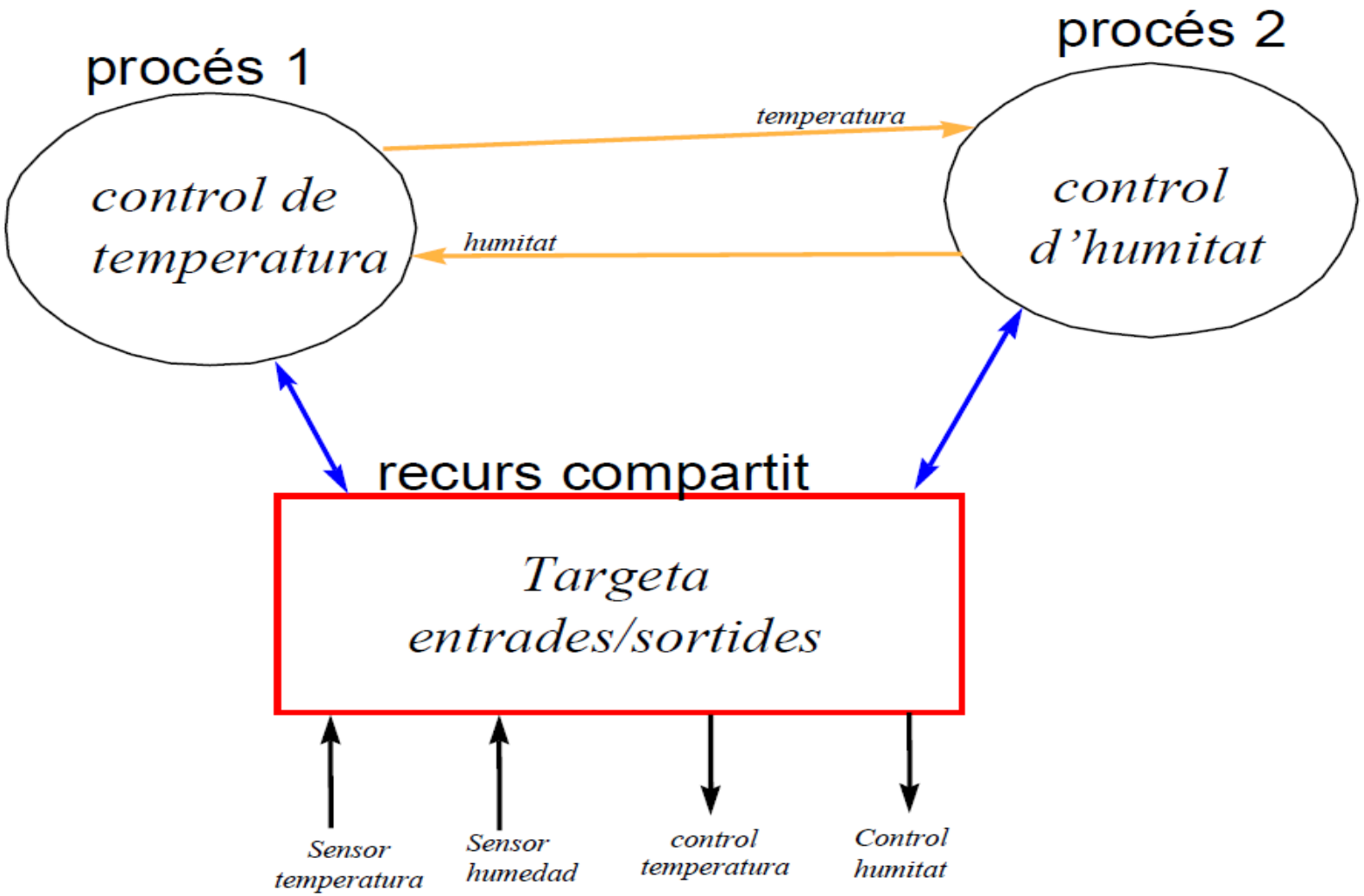
Comunicació i sincronització

- Les majors dificultats associades a la programació concurrent es deriven de la interacció entre processos. El correcte funcionament d'un programa concurrent depèn críticament de la sincronització i comunicació entre els processos.
- **La sincronització**: és el compliment de certes condicions en l'ordre d'execució de les accions de diferents processos.

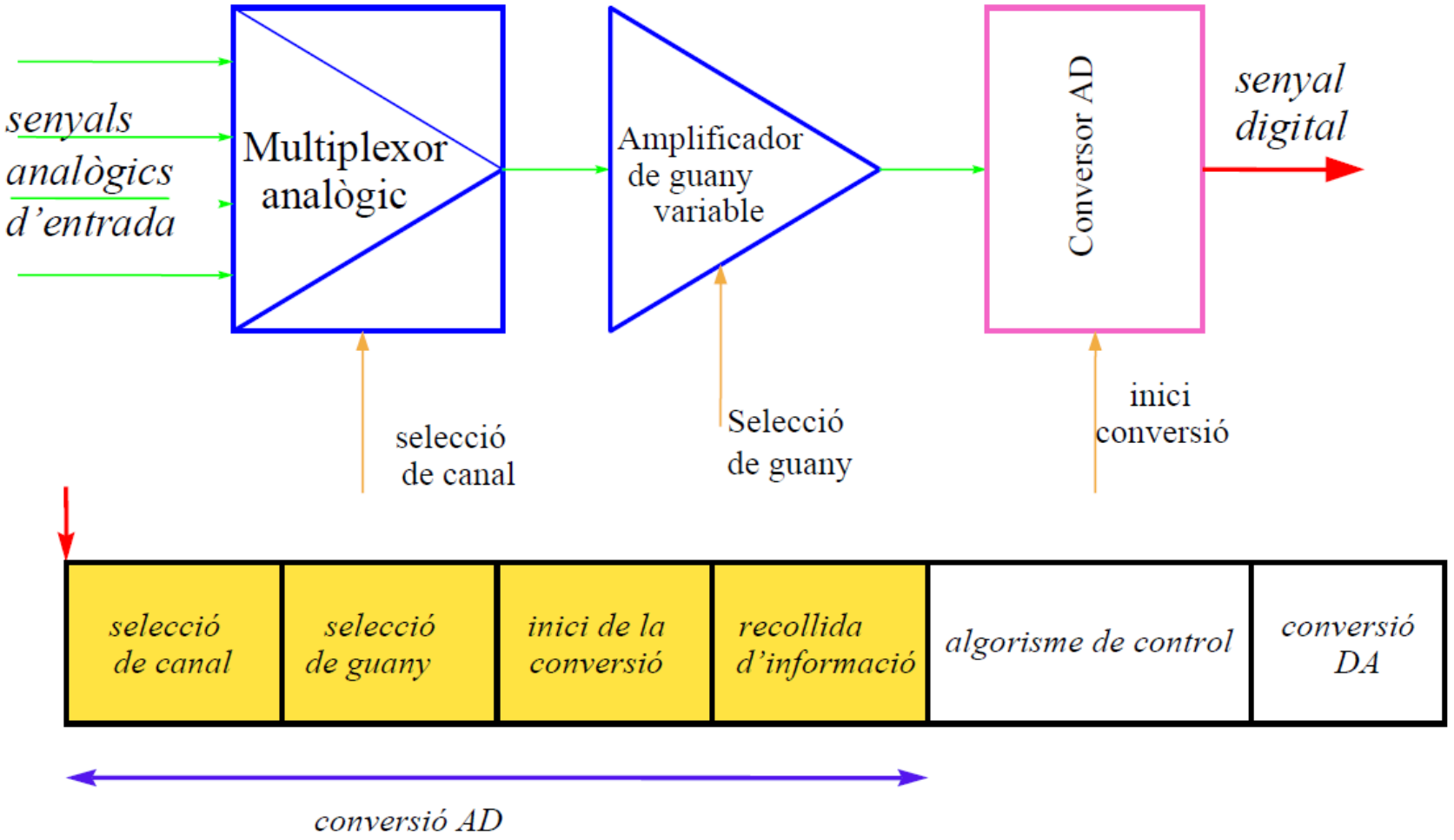
  Es pot entendre com el control de l'execució de 2 o més processos interactuant de manera que realitzin o no la mateixa operació simultàniament.
- **La comunicació:** és el pas d'informació d'un procés a un altre.

Ambdós conceptes estan units, ja que algunes formes de comunicació requereixen sincronització, i la sincronització pot ser considerada com una comunicació sense continguts.

Comunicació i sincronització entre processos



Comunicació i sincronització entre processos



Comunicació i sincronització entre processos

Procés 1

```
mentre no_final fer
    canal = 0;
    guany = 2;
    Inici conversió;
    Esperar final conversió;
    temp = Recollir informació;
    C_temp = f( temp, humi)
    Conversió DA
    Retard x
fmentre
```

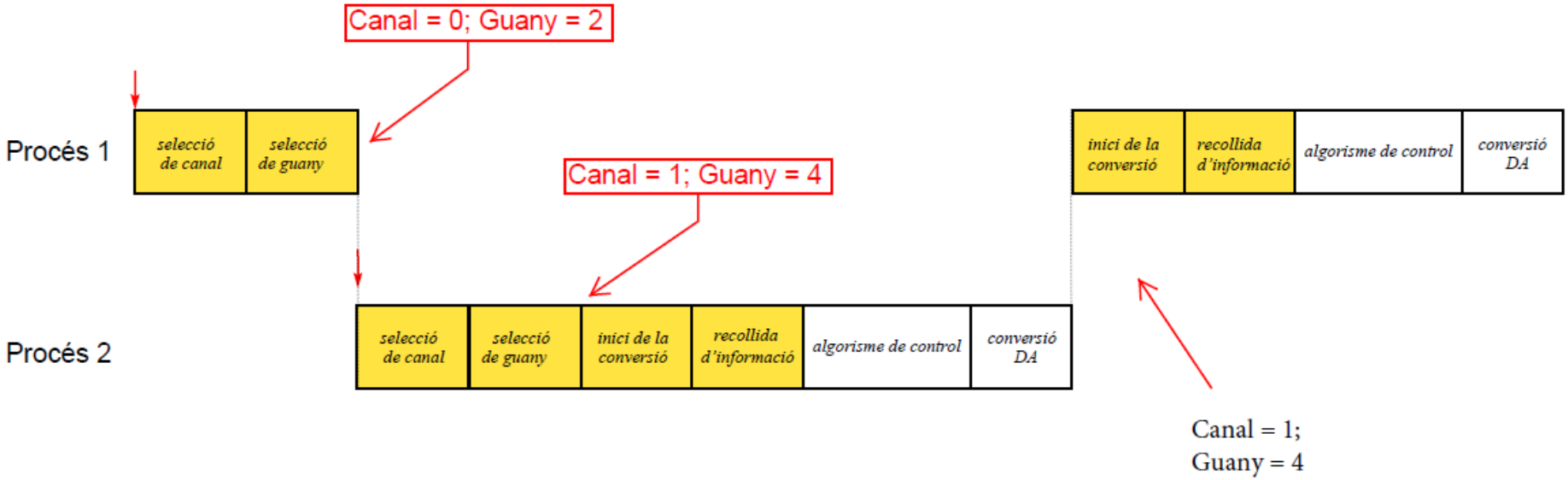
Procés 2

```
mentre no_final fer
    canal = 1;
    guany = 4;
    Inici conversió;
    Esperar final conversió;
    humi = Recollir informació;
    C_humi = g( temp, humi)
    Conversió DA
    Retard y
fmentre
```

Comunicació i sincronització entre processos

- Podem suposar la següent execució:

Procés 1 -> Control de temperatura (Canal = 0; Guany = 2)
Procés 2 -> Control d'humitat (Canal = 1; Guany = 4)



Comunicació: Les variables ***temp*** i ***humi*** serveixen per fer la comunicació entre els dos processos. En un procés es modifica i a l'altre es llegeix. No dona problemes de concurrència.

Sincronització: Hi ha una competència pels recursos????

Comunicació i sincronització entre processos

- **Operació atòmica** : conjunt d'instruccions que s'han d'executar de forma consecutiva sense interrupcions.
- Quan els recursos s'han d'accedir mitjançant operacions atòmiques es diu que s'han d'accedir en **exclusió mútua**.
- El tros de codi que s'ha d'executar en exclusió mútua rep el nom de **secció crítica**.

Sincronització entre processos

- Exclusió mútua

var mutex: semàfor; (* inicialitzat a 1 *)

```
procés P1;
      X;
      wait (mutex);
            Y; % Secció critica
      signal (mutex);
      Z;
end P1;
```

```
procés P2;
      A;
      wait (mutex);
            B; % Secció critica
      signal (mutex);
      C;
end P2;
```

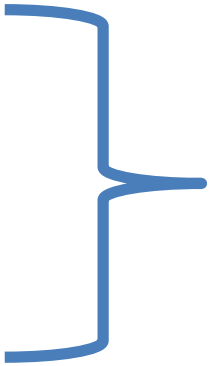
Incrementa mutex

Si mutex val 1 ->
continua i decrementa mutex
Si mutex val zero ->
S'espera

Comunicació i sincronització entre processos

Procés 1

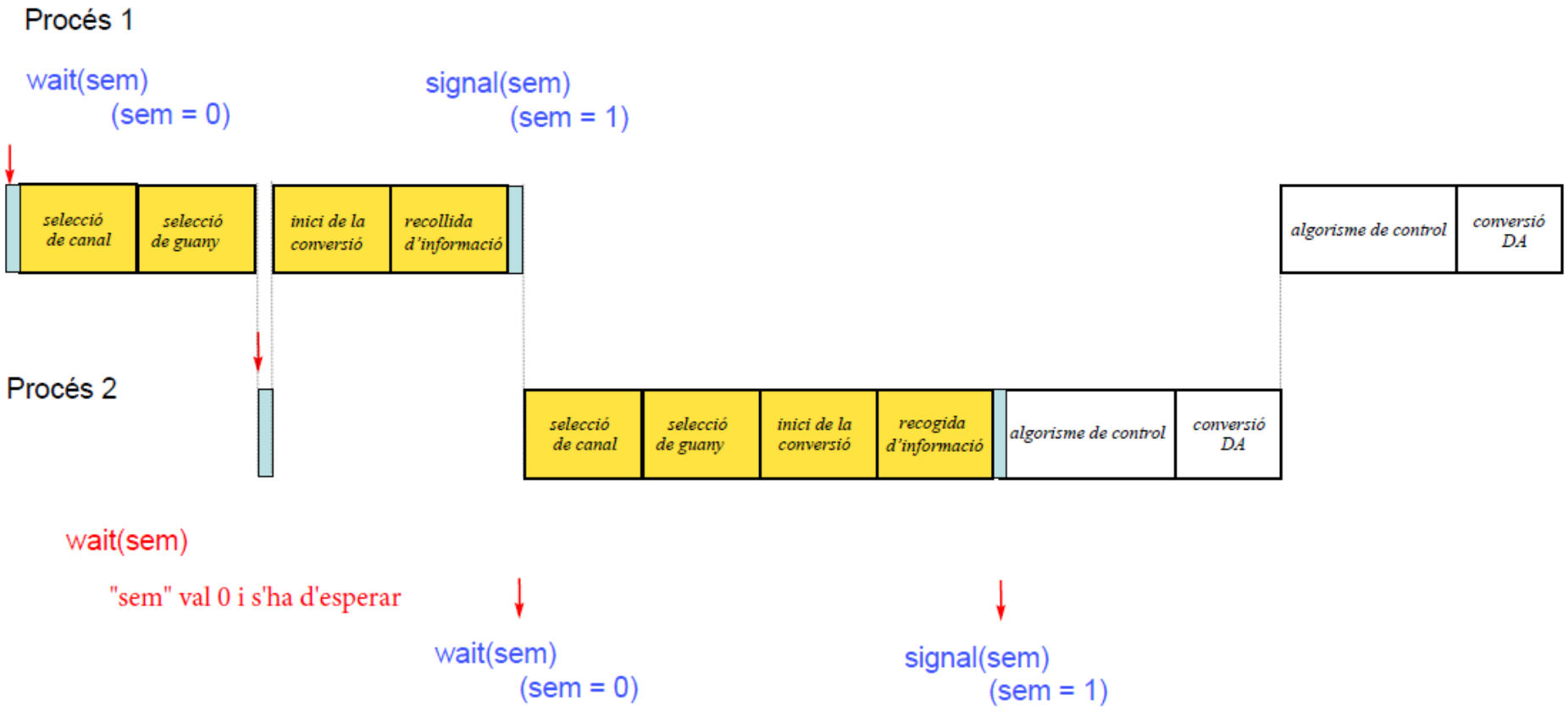
```
mentre no_final fer
    wait (sem);
        canal = 0;
        guany = 2;
        Inici conversió;
        Esperar final conversió;
        temp = Recollir informació;
    signal (sem);
    C_temp = f( temp, humi)
    Conversió DA
    retard
fmentre
```



Secció critica

Comunicació i sincronització entre processos

- Podem suposar la següent execució:



Comunicació i sincronització entre processos

- El correcte funcionament del sistema depèn de la sincronització i la comunicació entre els diferents processos.
  - ✓Sincronització : Emprada per indicar que en certs punts del codi es compleixen algunes restriccions.
  - ✓Comunicació: El pas d'informació entre diferents processos.
- La comunicació i la sincronització estan lligades. Ja que el pas d'informació requereix una sincronització.
  - La comunicació -> en variables compartides o en el pas de missatges.
  - La sincronització -> en espera activa i semàfors

Sincronització entre processos

- Espera activa
  - Una manera simple de sincronització és activar o desactivar indicadors (flags)
  - Encara que pot funcionar be en alguns casos no està clar que funcioni bé si no es disposa d'accions atòmiques (*test & set*)
  - El mecanismes d'espera activa són ineficients i poden donar lloc a càrregues molt grans al sistema (ex. *Multiprocessor, memory bus or network*)

Sincronització entre processos

- Semàfors
  - Un semàfor (S) és una variable sencera sense signe que es pot modificar mitjançant dos procediments P (or *WAIT*) i V (or SIGNAL)
  - WAIT(S): Si el valor de S és > 0 aleshores es decrementa el seu valor en 1; en cas contrari es retarda l'execució fins que S > 0 (i aleshores decrementa el seu valor).
  - SIGNAL(S):Incrementa el valor de S en una unitat.
  - WAIT i SIGNAL són atòmiques (indivisible). Dos processos que executen simultàniament WAIT sobre el mateix semàfor no es poden interferir.

Sincronització entre processos

El procés 1 (P1) per poder executar "Y", necessita que s'hagi executat "A" del procés 2 (P2)

var consyn : semàfor (* inicialització a 0 *)

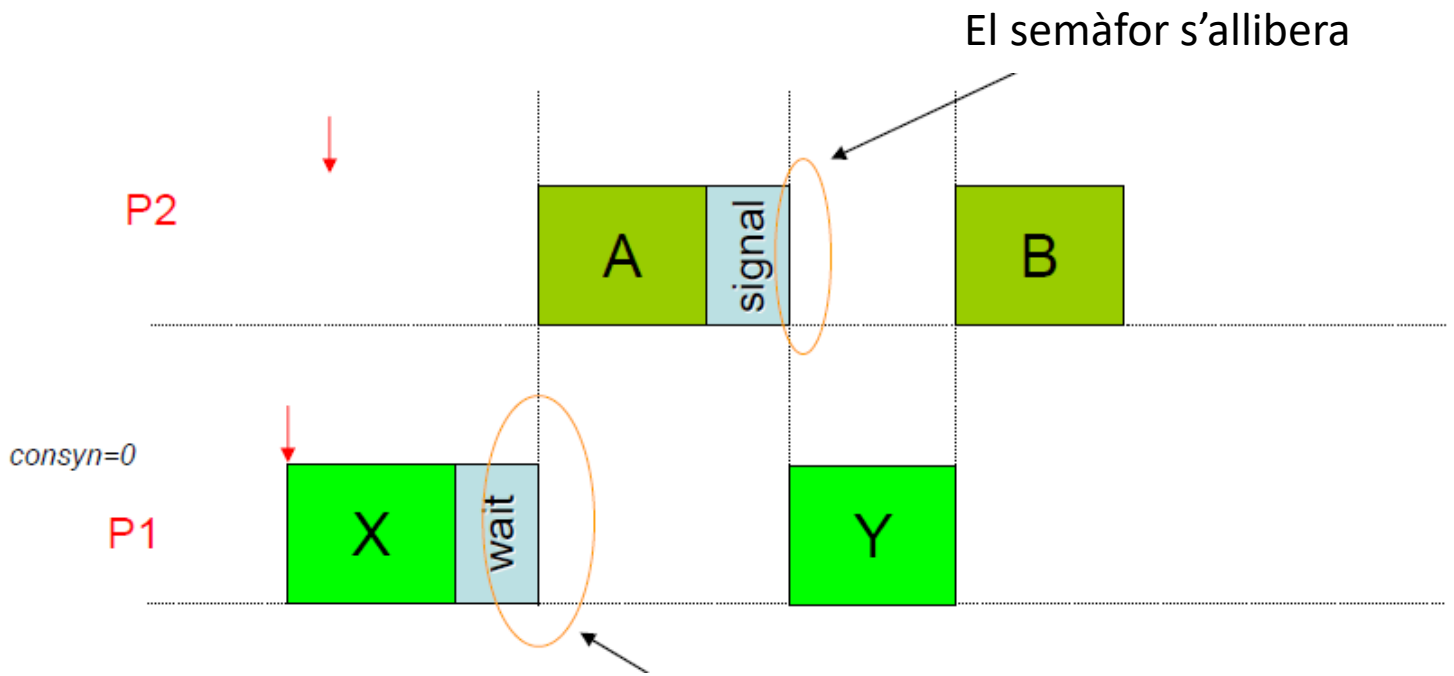
```
proces P1;
   (* Procés que espera *)
   X;
   wait (consyn)
   Y;
end P1;
```

Espera que s'activi "consyn".

```
proces P2;
   (* Procés que inicia *)
   A;
   signal (consyn)
   B;
end  P2;
```

Activa "consyn" (++)

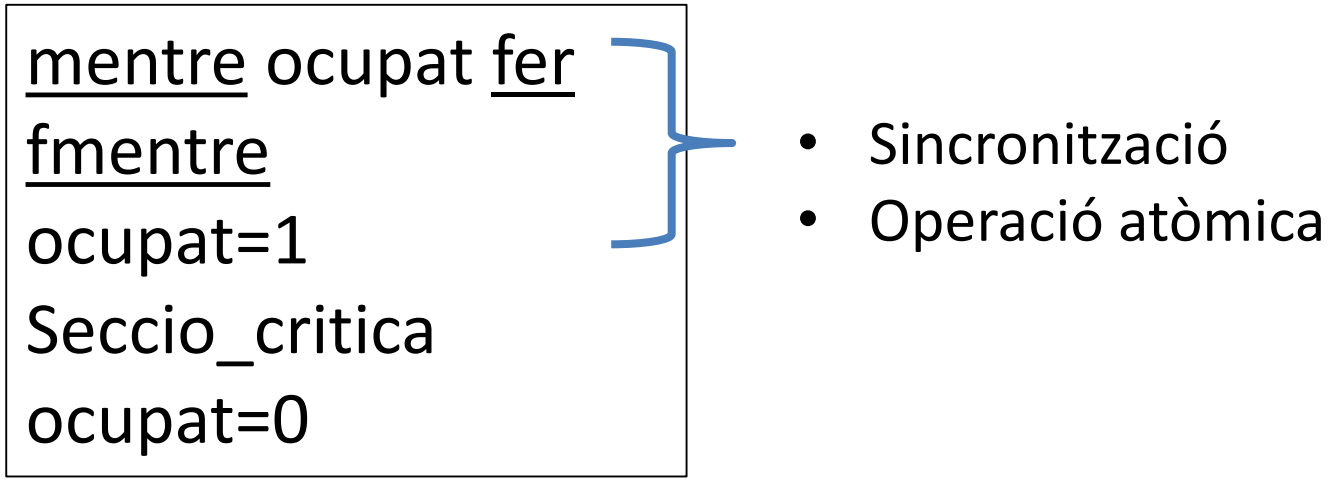
Sincronització entre processos



Seguint aquest plantejament es pot garantir que el bloc A s'executarà sempre abans que el Y.

Comunicació entre processos

- Variables compartides
  - L'ús de variables compartides sense sincronització és problemàtic i insegur.
  - La part del codi que accedeix a les variables compartides s'ha d'accedir en exclusió mútua



Comunicació entre processos

- Pas de missatges
  - Un missatge és una informació estructurada que envia un procés a un altre mitjançant un canal de comunicació.
  - El pas de missatges és imprescindible en sistemes distribuïts ja que no hi ha recursos directament compartits per intercanviar informació entre els processos.
  - És una tècnica utilitzada en la programació concurrent per aportar sincronització entre processos i permetre l'exclusió mútua, de manera similar a com es fa amb els semàfors, monitors, etc.
  - No precisa de memòria compartida, per la qual cosa és molt important en la programació per a sistemes distribuïts.
  - Els elements principals que intervenen en el pas de missatges són el procés que envia, el que el rep i el missatge.
  - Hi ha dos tipus de passos de missatges:
    - Asíncron: El procés que envia no espera que sigui rebut el missatge enviat. Aquests pot ser que tinguin una bústia per mantenir els missatges que s'han enviat prèviament i no han estat rebuts encara.
    - Síncron: El procés que envia el missatge, espera que aquest sigui rebut, abans de generar i enviar un altre missatge.