

A decorative frame consisting of two thick, dark brown L-shaped lines. One line starts at the top-left corner and extends horizontally to the right, then vertically down. The other line starts at the bottom-right corner and extends horizontally to the left, then vertically up. They meet at the center, framing the text.

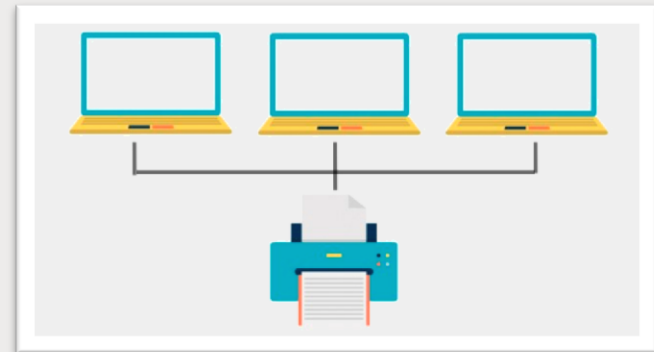
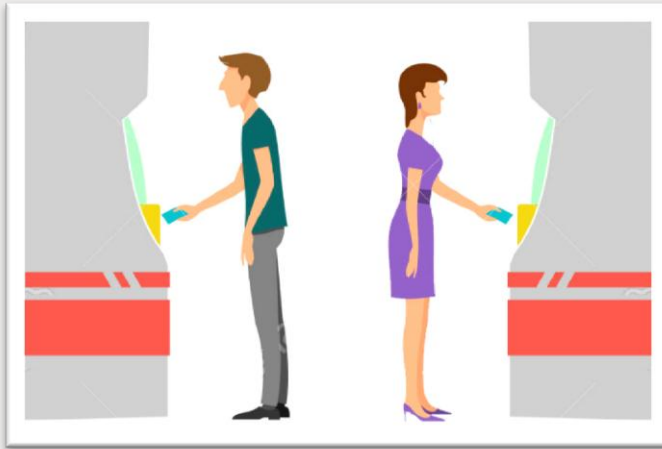
# TALLER DE PROGRAMACIÓN

Programación Concurrente

# ORGANIZACIÓN

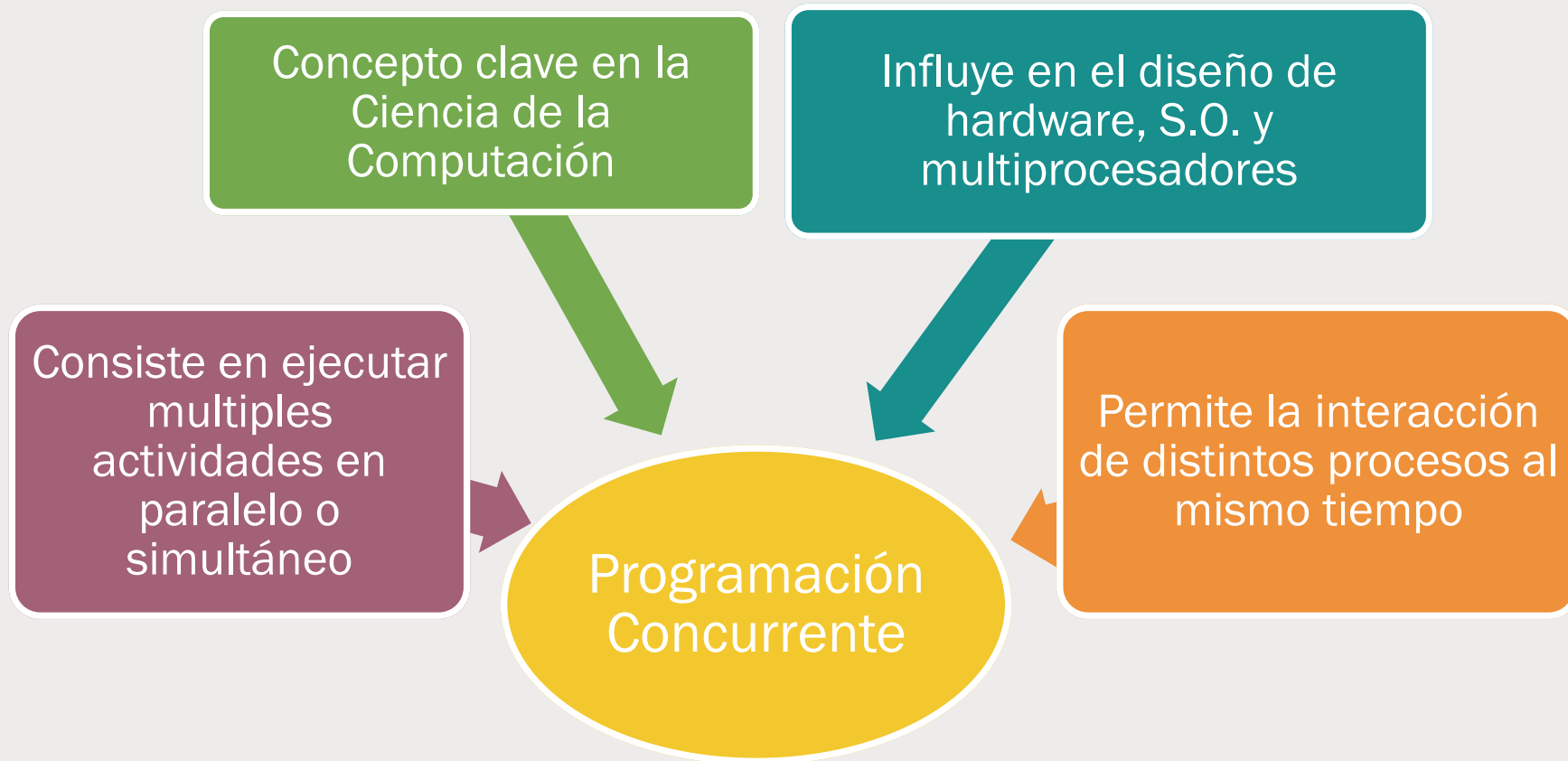


# PROGRAMACIÓN CONCURRENTE



# MOTIVACIÓN

## ¿Qué es la programación concurrente?

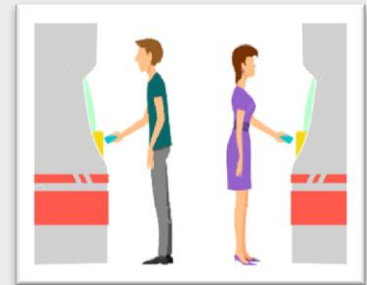


# MOTIVACIÓN

*¿Dónde está la concurrencia?*



Diferentes páginas en un navegador web y varios usuarios accediendo a la misma página.



Varias personas accediendo a distintas o a la misma cuenta

El Sistema Operativo de la computadora



Reserva de:  
pasajes,  
hotel,  
etc

Un smartphone



# CONCURRENCIA

## Escenarios

```
01: Var
02:   i,a,n,t: longint;
03:   d: array [1..10000] of longint;
04:
05: Procedure BusqLineal(x: longint):
06:   var i: longint;
07:   begin
08:     for i:= 1 to n do
09:       if (x= d[i] AND #00FF) then
10:         writeln(output,d[i] shr 8,' ',x);
11:       end;
12:
13: Begin
14:   t:= 0;
15:   readln(input,n);
16:   for i:= 1 to n do
17:     begin
18:       readln(input,d[i],a);
19:       if (a>t) then t:= a;
20:       d[i]:= (d[i] shr 8) OR a;
21:     end;
22:   for i:= t downto 0 do
23:     BusqLineal(i);
24: End.
```



Programa  
Secuencial



# CONCURRENCIA

## Escenarios

Programa  
Concurrente

```
01: Var
02:   i,m,n,c: longint;
03:   d: array [1..10000] of longint;
04:
05: Procedure BusqLineal(x: longint);
06:   var i: longint;
07:   begin
08:     for i:= 1 to n do
09:       if (x= d[i] AND x<0) then
10:         writeln(output,d[i] shr 8,' ',x);
11:     end;
12:
13: Begin
14:   c:= 0;
15:   readln(input,n);
16:   for i:= 1 to n do
17:     begin
18:       readln(input,d[i],m);
19:       if (m<c) then c:= m;
20:       d[i]:= (d[i] shr 8) OR m;
21:     end;
22:   for i:= c downto 0 do
23:     BusqLineal(i);
24: End.
```



Un programa ejecutándose en una máquina dividido en partes tratando de acceder a un recurso compartido (ej. Impresora)

# CONCURRENCIA

## Escenarios

Programa  
Paralelo

```
01: Var
02:   i,m,n,c: longint;
03:   d: array [1..10000] of longint;
04:
05: Procedure BusqLineal(x: longint);
06:   var i: longint;
07:   begin
08:     for i:= 1 to n do
09:       if (x= d[i] AND #00FF) then
10:         writeln(output,d[i] shr 8,' ',x);
11:     end;
12:
13: Begin
14:   c:= 0;
15:   readln(input,n);
16:   for i:= 1 to n do
17:     begin
18:       readln(input,d[i],m);
19:       if (m<c) then c:= m;
20:       d[i]:= (d[i] shr 8) OR m;
21:     end;
22:   for i:= c downto 0 do
23:     BusqLineal(i);
24: End.
```



Un programa ejecutándose en **varias máquinas** tratando de acceder a un **recurso compartido** (ej. Impresora)



# CONCURRENCIA

## Ejemplo



SE TIENEN



**Automóviles:**

Procesos que se ejecutan



**Carriles y rutas:**

Múltiples procesadores

Los **automóviles** deben **sincronizarse** para no chocar

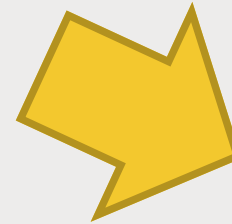
**Objetivo:** examinar los tipos de autos (procesos), trayecto a recorrer (programas), caminos (hardware), y reglas (comunicación y sincronización).

# CONCURRENCIA



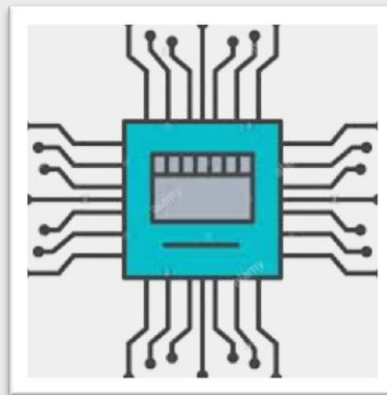
Concurrencia

Es la característica de los sistemas que indica que múltiples procesos/tareas pueden ser ejecutados al mismo tiempo y pueden cooperar y coordinarse para cumplir la función del sistema.

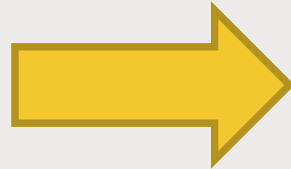


CAMBIO EN EL  
SOFTWARE

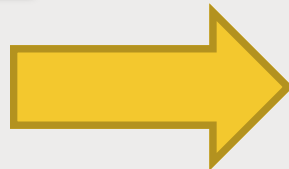
*Y el  
hardware?*



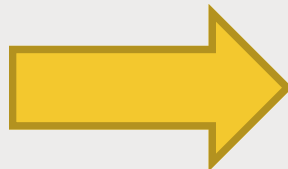
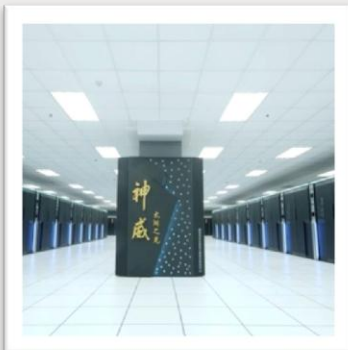
# CONCURRENCIA



1 sólo núcleo de procesamiento  
(1980)



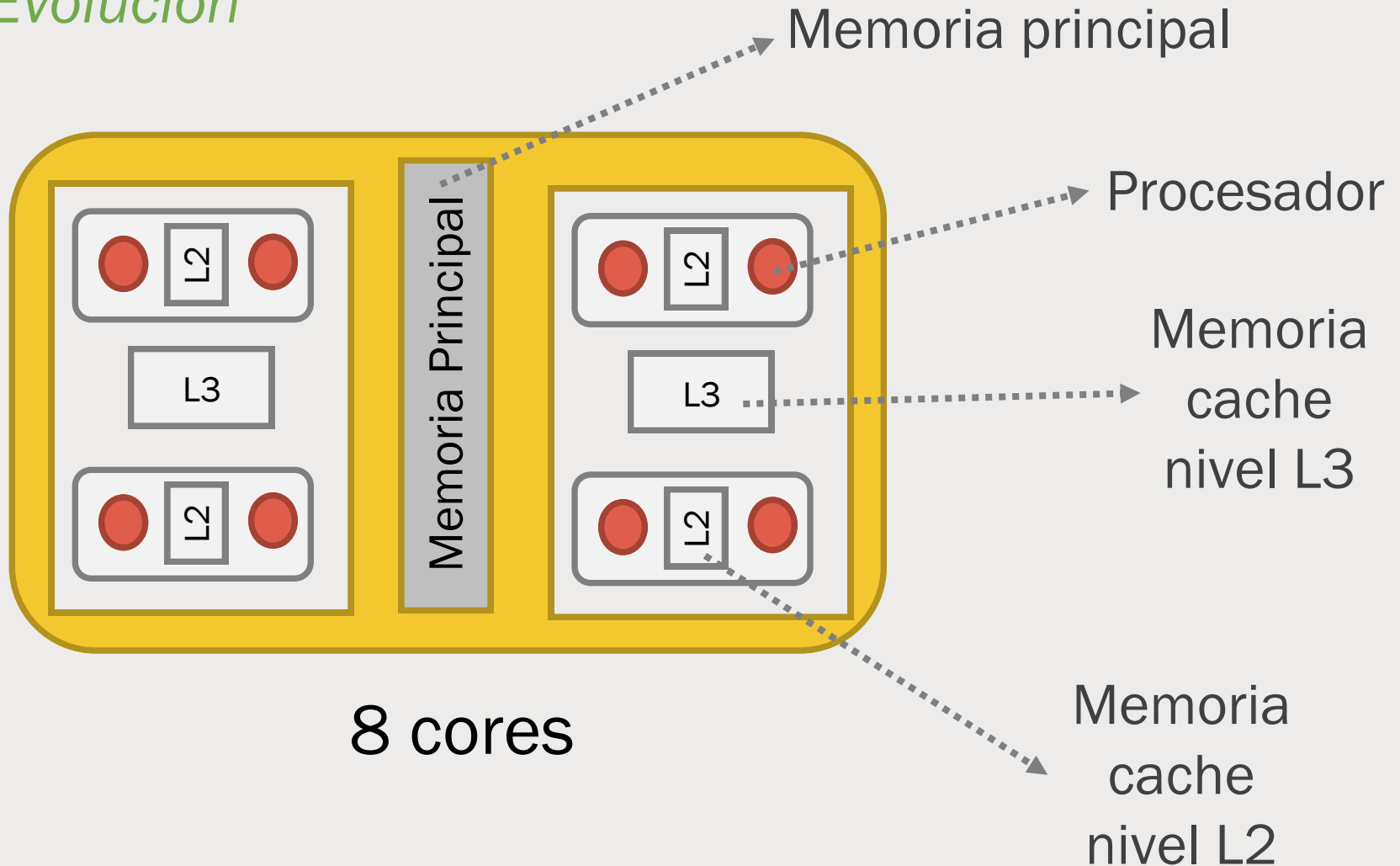
Se agregan placas que agrupan  
núcleos (2, 4, 8, etc )  
(2000)



La 1<sup>era</sup> computadora del Top500  
10 millones de núcleos!!!!  
(2017)

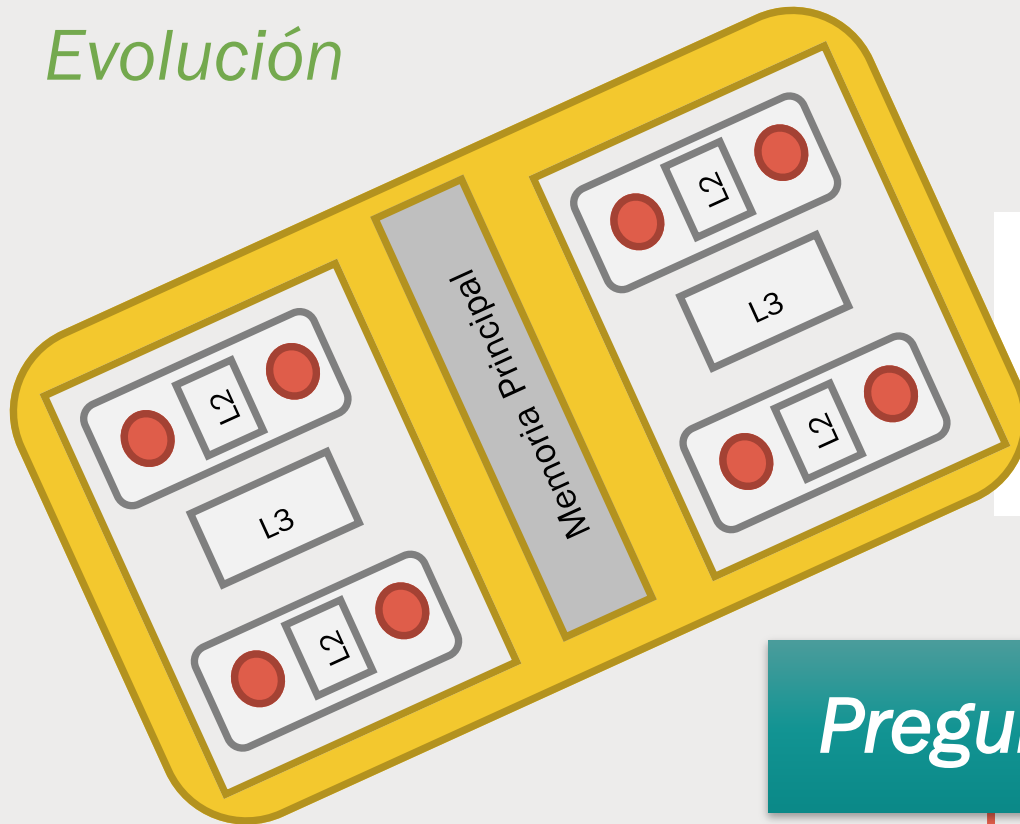
# CONCURRENCIA

## *Evolución*



# CONCURRENCIA

## Evolución



PARA PODER EXPLOTAR ESTE  
HARDWARE ES NECESARIO  
PROGRAMAR PROCESOS  
CONCURRENTES !

## Preguntas

*¿Cómo se  
maneja la  
conurrencia?*

*¿Cómo  
interactúan los  
procesos?*

*¿Cómo armamos  
un programa  
correcto?*

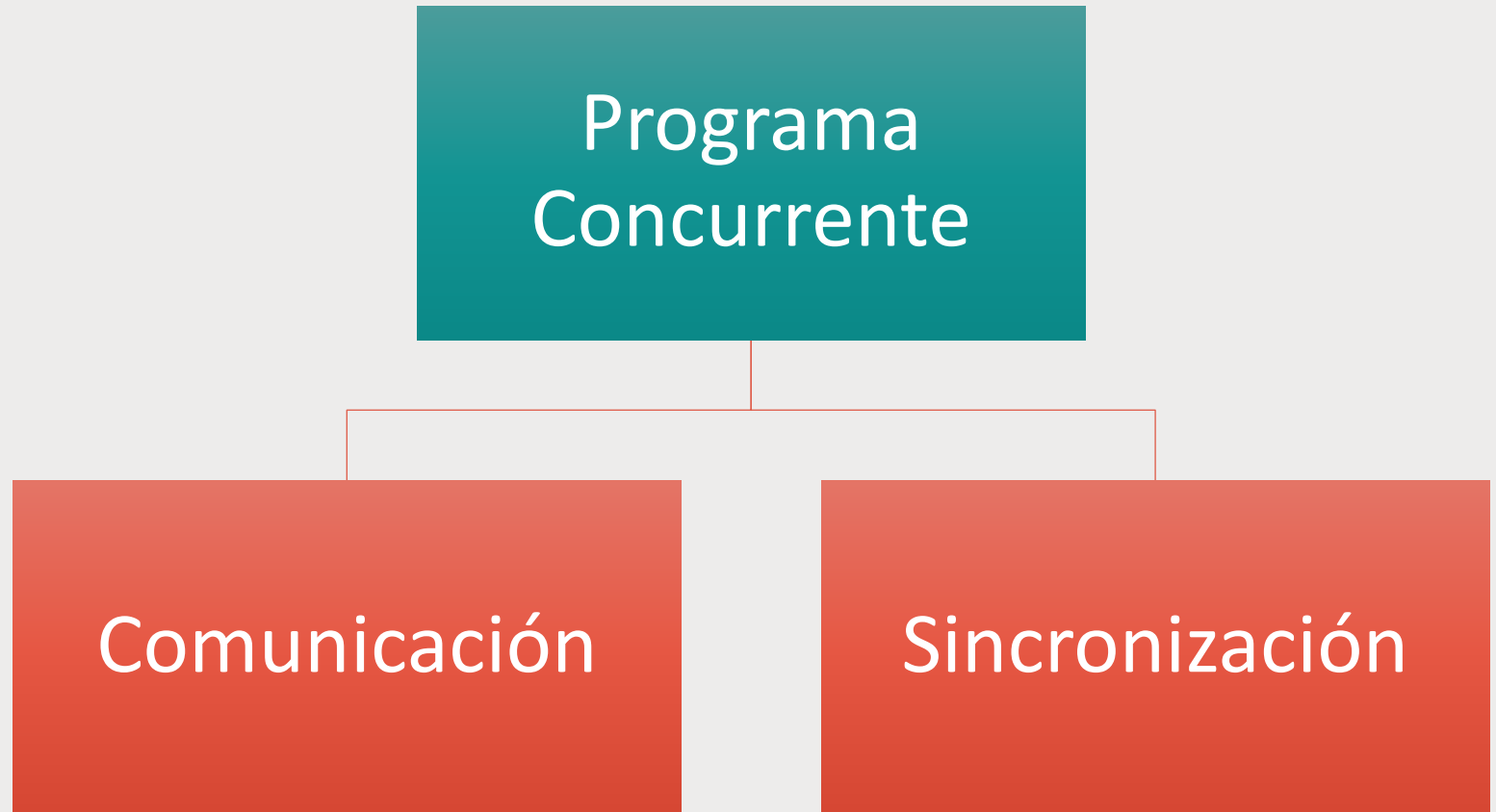
# PROGRAMACION CONCURRENTES

Los procesos concurrentes tendrán necesidad de **comunicarse** información.

Además, será necesario en ocasiones detener a un proceso hasta que se produzca un determinado evento o se den ciertas condiciones **sincronización**

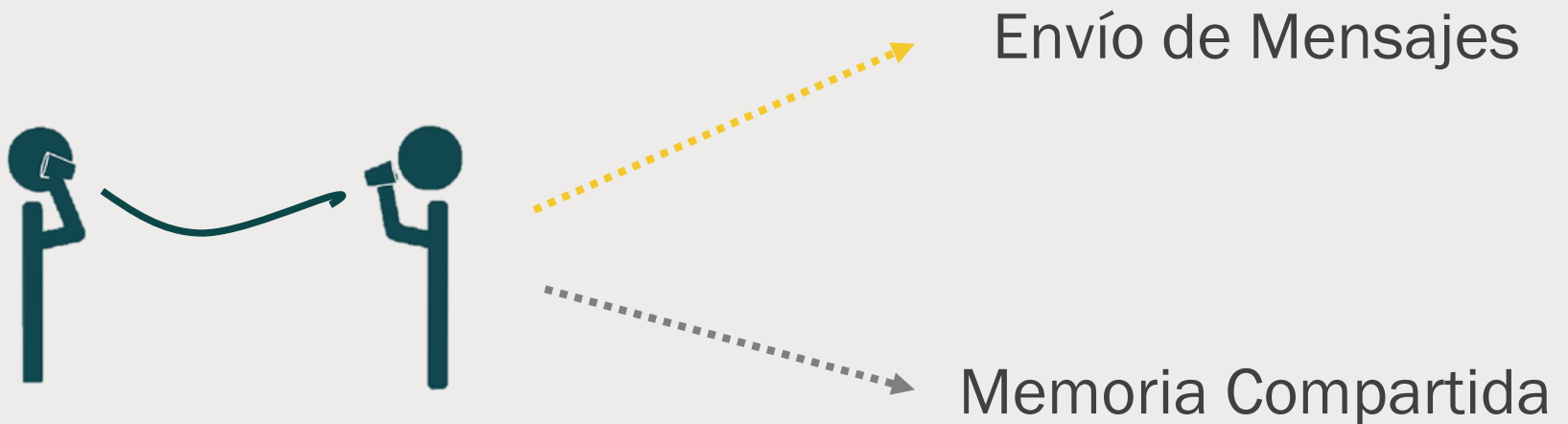
Los **lenguajes concurrentes** deben proporcionar mecanismos de sincronización y comunicación.

# PROGRAMACION CONCURRENTES



# PROGRAMACION CONCURRENTE

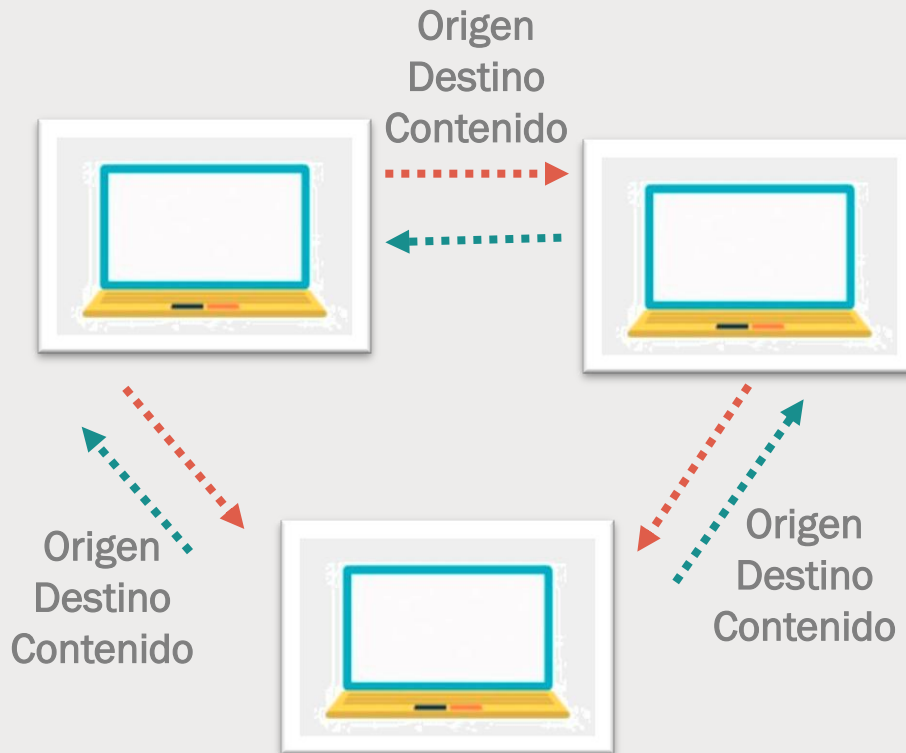
## *Comunicación*





# PROGRAMACION CONCURRENTES

## Comunicación – 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 transmitir mensajes.

**ENVIAR Y RECIBIR**

# PROGRAMACION CONCURRENTES

## Comunicación – PASAJE DE MENSAJES - Ejemplo

```
MPI_Send (buff, 128, MPI_CHAR, 1 , 0, MPI_COMM_WORLD);
```



**En CMRE**

```
EnviarMensaje (dato, robotDestino)  
RecibirMensaje (dato, robotOrigen)
```

# PROGRAMACION CONCURRENTE

## Comunicación – PASAJE DE MENSAJES -


Envío de Mensajes



Sincrónico

Asincrónico

Recepción de Mensajes



Sincrónico

Asincrónico

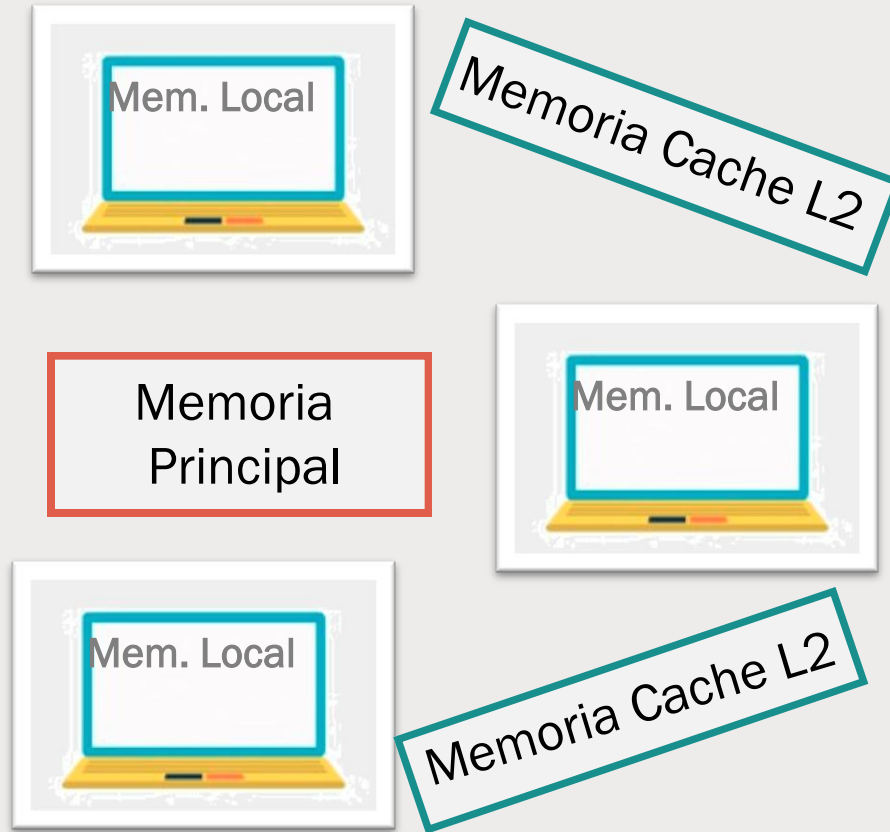
**En CMRE**

Envío Asincrónico

Recepción Sincrónica

# PROGRAMACION CONCURRENTE

## Comunicación – MEMORIA COMPARTIDA

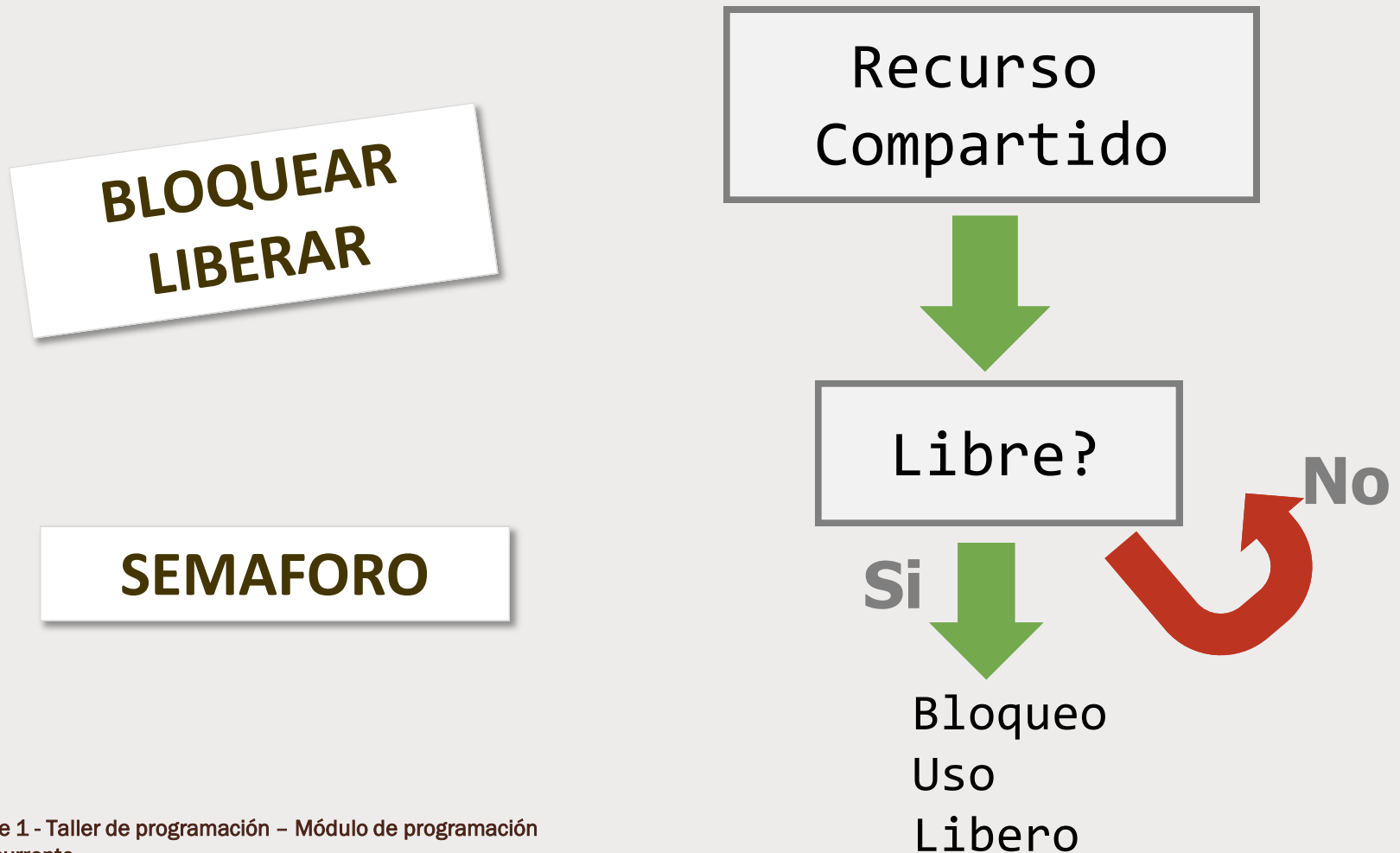


- Los procesos intercambian información sobre la memoria compartida o actúan coordinadamente sobre datos residentes en ella.
- Lógicamente no pueden operar simultáneamente sobre la memoria compartida, lo que obliga a bloquear y liberar el acceso a la memoria.
- La solución más elemental es una variable de control que habilite o no el acceso de un proceso a la memoria compartida.

**BLOQUEAR  
LIBERAR**

# PROGRAMACION CONCURRENTE

## Comunicación – MEMORIA COMPARTIDA



# PROGRAMACION CONCURRENTES

## Comunicación – MEMORIA COMPARTIDA - Ejemplo

`P(variable semáforo);`    `V(variable semáforo);`



Protege un recurso



Libera un recurso

**En CMRE**

`BloquearEsquina (avenida,calle)`  
`LiberarEsquina (avenida,calle)`

# PROGRAMACION CONCURRENTE



Suponga que una pareja comparte su cuenta bancaria. En un momento los dos integrantes de la pareja van a un cajero y extraen 1000 pesos. Es correcto el siguiente código?

Variable compartida **saldo**

Integrante 1:

```
{  
  accede a la cuenta  
  saldo:= saldo - 1000;  
}
```

Integrante 2:

```
{  
  accede a la cuenta  
  saldo:= saldo - 1000;  
}
```

# PROGRAMACION CONCURRENTE



Suponga que una pareja comparte su cuenta bancaria. En un momento los dos integrantes de la pareja van a un cajero y extraen 1000 pesos. Es correcto el siguiente código?

Variable compartida **saldo**

Se puede mejorar?

```
Integrante 1:  
{ P(saldo)  
  accede a la cuenta  
  saldo:= saldo - 1000;  
  V(saldo)  
}
```

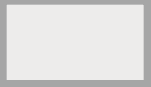
```
Integrante 2:  
{ P(saldo)  
  accede a la cuenta  
  saldo:= saldo - 1000;  
  V(saldo)  
}
```

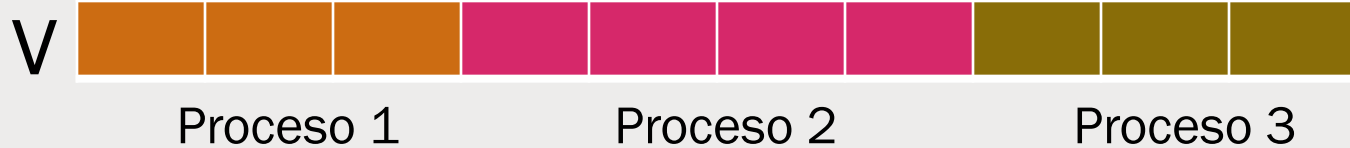


# PROGRAMACION CONCURRENTE



En un programa existen 3 procesos, un arreglo de longitud M y un valor N y se quiere calcular cuantas veces aparece el valor N en el arreglo.

  
cont



Variable compartida **cont** y **V**

Es correcto?

```
Proceso 1:  
{inf:=...; sup:= ...;  
  
  for i:= inf to sup do  
    if v[i] = N then  
      cont:= cont + 1;  
}
```

```
Proceso 2:  
{inf:=...; sup:= ...;  
  
  for i:= inf to sup do  
    if v[i] = N then  
      cont:= cont + 1;  
}
```

```
Proceso 3:  
{inf:=...; sup:= ...;  
  
  for i:= inf to sup do  
    if v[i] = N then  
      cont:= cont + 1;  
}
```

# PROGRAMACION CONCURRENTE



En un programa existen 3 procesos, un arreglo de longitud M y un valor N y se quiere calcular cuantas veces aparece el valor N en el arreglo.

Variable compartida **cont** y **V**

Es correcto?

Se puede mejorar?

```
Proceso 1:
{inf:=...; sup:= ...;
  for i:= inf to sup do
    P(variable)
    if v[i] = N then
      cont:= cont + 1;
    V(variable)
}
```

```
Proceso 2:
{inf:=...; sup:= ...;
  for i:= inf to sup do
    P(variable)
    if v[i] = N then
      cont:= cont + 1;
    V(variable)
}
```

```
Proceso 3:
{inf:=...; sup:= ...;
  for i:= inf to sup do
    P(variable)
    if v[i] = N then
      cont:= cont + 1;
    V(variable)
}
```

# PROGRAMACION CONCURRENTE



En un programa existen 3 procesos, un arreglo de longitud M y un valor N y se quiere calcular cuantas veces aparece el valor N en el arreglo.

Variable compartida **cont** y **V**

Se puede mejorar?

Proceso 1:

```
{inf:=...; sup:= ...;
  for i:= inf to sup do
    if v[i] = N then
      P(variable)
      cont:= cont + 1;
      V(variable)
}
```

Proceso 2:

```
{inf:=...; sup:= ...;
  for i:= inf to sup do
    if v[i] = N then
      P(variable)
      cont:= cont + 1;
      V(variable)
}
```

Proceso 3:

```
{inf:=...; sup:= ...;
  for i:= inf to sup do
    if v[i] = N then
      P(variable)
      cont:= cont + 1;
      V(variable)
}
```

# PROGRAMACION CONCURRENTE



En un programa existen 3 procesos, un arreglo de longitud M y un valor N y se quiere calcular cuantas veces aparece el valor N en el arreglo.

Variable compartida **cont** y **V**

Proceso 1:

```
{inf:=...; sup:= ...; cant
```

```
  for i:= inf to sup do  
    if v[i] = N then  
      cant := cant + 1
```

```
  P(variable)  
  cont:= cont + cant;  
  V(variable)
```

```
}
```

Proceso 2:

```
{inf:=...; sup:= ...; cant
```

```
  for i:= inf to sup do  
    if v[i] = N then  
      cant := cant + 1
```

```
  P(variable)  
  cont:= cont + cant;  
  V(variable)
```

```
}
```

Proceso 3:

```
{inf:=...; sup:= ...; cant
```

```
  for i:= inf to sup do  
    if v[i] = N then  
      cant := cant + 1
```

```
  P(variable)  
  cont:= cont + cant;  
  V(variable)
```

```
}
```

# Ambiente CMRE

*FIN*