

## Concurrencia de Procesos: Exclusión Mutua y Sincronización

Eloy Anguiano Rey  
eloy.anguiano@uam.es

Rosa M. Carro  
rosa.carro@uam.es

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

Cartagena99

# Introducción

## Elementos a tener en cuenta

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

### Introducción

#### Elementos a tener en cuenta

Términos clave  
Dificultades  
Ejemplos de  
problemas

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

### Afecta a ..

- ... la comunicación entre procesos.
- ... la compartición y competencia por los recursos.
- ... la sincronización de la ejecución de varios procesos.
- ... la asignación del tiempo de procesador a los procesos.

### Presente en ...

- ... la ejecución de múltiples aplicaciones:
  - Multiprogramación
- ... las aplicaciones estructuradas:

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

# Introducción

## Términos clave

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

### Introducción

Elementos a tener en  
cuenta

### Términos clave

Dificultades

Ejemplos de  
problemas

### Memoria

compartida en  
UNIX

### Sincronización

Exclusión mutua

**Sincronización:** Los procesos coordinan sus actividades

**Sección crítica:** Región de código que sólo puede ser accedida por un proceso simultáneamente (variables compartidas).

**Exclusión mutua:** Sólo un proceso puede estar en sección crítica accediendo a recursos compartidos

**Interbloqueo:** Varios procesos, todos tienen algo que otros esperan, y a su vez esperan algo de los otros.

**Círculo vicioso:** Procesos cambian continuamente de estado como respuesta a cambios en otros procesos, sin que sea útil (ej: liberar recurso)

CLASES PARTICULARES, TUTORIAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP. 689 45 44 70

Cartagena99

# Introducción

## Dificultades con la concurrencia

La ejecución intercalada de procesos mejora rendimiento, pero ... **la velocidad relativa de los procesos no puede predecirse** puesto que depende de:

- Actividades de otros procesos
- Forma de tratar interrupciones
- Políticas de planificación

Pero esto implica que surgen dificultades

### Ejemplo

Hora	Mi compañera	Yo
3:00	Mira en la nevera <b>No hay leche</b>	
3:05	Sale hacia la tienda	
3:10	Entra en la tienda	Miro en la nevera <b>No hay leche</b>

CLASES PARTICULARES TUTORÍAS TÉCNICAS  
 LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
 ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
 CALL OR WHATSAPP. 689 45 44 70

Cartagena99

# Introducción

## Dificultades con la concurrencia

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

### Introducción

Elementos a tener en  
cuenta

Términos clave

**Dificultades**

Ejemplos de  
problemas

**Memoria  
compartida en  
UNIX**

**Sincronización**

**Exclusión mutua**

La imprevisibilidad de la velocidad relativa de los procesos implica que es difícil ...

- ... compartir recursos. Ej: orden de lecturas y escrituras.
- ... gestionar la asignación óptima de recursos. Ej: recursos asignados a un proceso y éste se bloquea, ¿recurso bloqueado?  $\Rightarrow$  posible interbloqueo
- ... detectar errores de programación (resultados no deterministas, no reproducibles)

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

# Introducción

## Ejemplos de problemas

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

### Introducción

Elementos a tener en  
cuenta  
Términos clave  
Dificultades

### Ejemplos de problemas

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

- Supóngase que se lanzan dos procesos idénticos con la siguiente estructura:

```
1 void echo()  
2 {  
3     ent = getchar();  
4     sal = ent;  
5     putchar(sal);  
6 }
```

- Cuando se ejecutan los dos simultáneamente la ejecución puede ser la siguiente:

1  
2

```
...  
ent = getchar();
```

1  
2

```
...  
...
```

CLASES PARTICULARES Y TEORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

# Introducción

## Ejemplos de problemas

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

### Introducción

Elementos a tener en  
cuenta

Términos clave  
Dificultades

Ejemplos de  
problemas

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Supóngase la ejecución de estos dos procesos con variables compartidas

Proceso A

```
1 for(i=1 to 5) do {  
2   x=x+1;  
3 }
```

Proceso B

```
1 for(j=1 to 5) do {  
2   x=x+1;  
3 }
```

con las siguientes condiciones:

- Valor inicial  $x=0$ .
- Se comparten todas las variables.
- La operación de incremento se realiza en tres instrucciones atómicas:

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

# Introducción

## Ejemplos de problemas

### Caso mínimo

i	inst	BCPa	Acc	X	BCPb	inst	j
1	LD Acc		0	0			
		0	0	0		LD Acc	1

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

#### Introducción

Elementos a tener en  
cuenta

Términos clave

Dificultades

Ejemplos de  
problemas

#### Memoria

compartida en  
UNIX

#### Sincronización

Exclusión mutua



Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
 LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
 ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
 CALL OR WHATSAPP: 689 45 44 70



# Introducción

## Ejemplos de problemas

### Caso mínimo

i	inst	BCPa	Acc	X	BCPb	inst	j
1	LD Acc		0	0			
		0	0	0		LD Acc	1
		0	1	0		Acc++	
		0	1	1		SV Acc	
		0	1	1		LD Acc	2
		0	2	1		Acc++	
		0	2	2		SV Acc	
		0	2	2		LD Acc	3
		0	3	2		Acc++	
		0	3	3		SV Acc	
		0	3	3		LD Acc	4

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
 LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
 ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
 CALL OR WHATSAPP: 689 45 44 70

Cartagena99

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

#### Introducción

Elementos a tener en  
cuenta

Términos clave

Dificultades

Ejemplos de  
problemas

Memoria

compartida en

UNIX

Sincronización

Exclusión mutua

# Introducción

## Ejemplos de problemas

### Caso mínimo

i	inst	BCPa	Acc	X	BCPb	inst	j
1	LD Acc		0	0			
		0	0	0		LD Acc	1
		0	1	0		Acc++	
		0	1	1		SV Acc	
		0	1	1		LD Acc	2
		0	2	1		Acc++	
		0	2	2		SV Acc	
		0	2	2		LD Acc	3
		0	3	2		Acc++	
		0	3	3		SV Acc	
		0	3	3		LD Acc	4

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
 LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
 ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
 CALL OR WHATSAPP: 689 45 44 70

Cartagena99

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

#### Introducción

Elementos a tener en  
cuenta

Términos clave

Dificultades

Ejemplos de  
problemas

Memoria

compartida en

UNIX

Sincronización

Exclusión mutua

# Introducción

## Ejemplos de problemas

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

### Caso mínimo

i	inst	BCPa	Acc	X	BCPb	inst	j
1	LD Acc		0	0			
		0	0	0		LD Acc	1

#### Introducción

Elementos a tener en  
cuenta

Términos clave  
Dificultades

Ejemplos de  
problemas

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua



Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS/  
CALL OR WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

# Introducción

## Ejemplos de problemas

### Caso mínimo

i	inst	BCPa	Acc	X	BCPb	inst	j
		1	1	1	4	LD Acc	5
2	LD Acc	1	1	1	1		
	Acc++	1	2	1	1		
	SV Acc	1	2	2	1		
3	LD Acc	1	2	2	1		
	Acc++	1	3	2	1		
	SV Acc	1	3	3	1		
4	LD Acc	1	3	3	1		
	Acc++	1	4	3	1		
	SV Acc	1	4	4	1		
5	LD Acc	1	4	4	1		

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
 LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
 ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
 CALL OR WHATSAPP: 689 45 44 70

Cartagena99

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

#### Introducción

Elementos a tener en  
cuenta

Términos clave

Dificultades

Ejemplos de  
problemas

Memoria

compartida en  
UNIX

Sincronización

Exclusión mutua

# Introducción

## Ejemplos de problemas

### Caso mínimo

i	inst	BCPa	Acc	X	BCPb	inst	j
		1	1	1	4	LD Acc	5
2	LD Acc	1	1	1	1		
	Acc++	1	2	1	1		
	SV Acc	1	2	2	1		
3	LD Acc	1	2	2	1		
	Acc++	1	3	2	1		
	SV Acc	1	3	3	1		
4	LD Acc	1	3	3	1		
	Acc++	1	4	3	1		
	SV Acc	1	4	4	1		
5	LD Acc	1	4	4	1		
	Acc++	1	5	4	1		

CLASES PARTICULARES TUTORÍAS TÉCNICAS  
 LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
 ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
 CALL OR WHATSAPP. 689 45 44 70

Cartagena99

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

#### Introducción

Elementos a tener en  
cuenta

Términos clave

Dificultades

Ejemplos de  
problemas

Memoria

compartida en  
UNIX

Sincronización

Exclusión mutua

# Introducción

## Ejemplos de problemas

### Caso máximo

i	inst	BCPa	Acc	X	BCPb	inst	j
			0	0		LD Acc	1
			1	0		Acc++	
			1	1		SV Acc	
			1	1		LD Acc	2
			2	1		Acc++	
			2	2		SV Acc	
			2	2		LD Acc	3
			3	2		Acc++	
			3	3		SV Acc	
			3	3		LD Acc	4
			4	3		Acc++	

CLASES PARTICULARES Y TUTORÍAS TÉCNICAS  
 LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
 ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
 CALL OR WHATSAPP. 689 45 44 70

Cartagena99

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

#### Introducción

Elementos a tener en  
cuenta

Términos clave

Dificultades

Ejemplos de  
problemas

Memoria

compartida en

UNIX

Sincronización

Exclusión mutua

# Introducción

## Ejemplos de problemas

### Caso máximo

i	inst	BCPa	Acc	X	BCPb	inst	j
1	LD Acc		5	5			
	Acc++		6	5			
	SV Acc		6	6			
2	LD Acc		6	6			
	Acc++		7	6			
	SV Acc		7	7			
3	LD Acc		7	7			
	Acc++		8	7			
	SV Acc		8	8			
4	LD Acc		8	8			
	Acc++		9	8			
	SV Acc		9	9			

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
 LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
 ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
 CALL OR WHATSAPP: 689 45 44 70

Cartagena99

# Introducción

## Ejemplos de problemas

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

### Introducción

Elementos a tener en  
cuenta

Términos clave  
Dificultades

Ejemplos de  
problemas

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

El proveedor produce información para el consumidor. La concurrencia se produce mediante el uso de buffers y variables compartidas (compartición de memoria) o mediante compartición de ficheros.



En el que se tienen las siguientes condiciones:

- Uno o más productores generan datos y los sitúan en un buffer

**CLASES PARTICULARES, TUTORIAS TÉCNICAS  
 LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
 ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
 CALL OR WHATSAPP. 689 45 44 70**

**Cartagena99**



# Introducción

## Ejemplos de problemas

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Compartido

```

1  #define N 100 /*mximo nmero de elementos */
2  int contador=0; /* contador del nmero de elementos disponibles */
3  typedef type item; /* definicin del producto/consumible */
4  item array[N]; /* array circular, con ndice en mdulo N (0..N-1) */
5  item *in=array; /* puntero a la siguiente posicin libre */
6  item *out=NULL; /* puntero primer elemento ocupado */

```

Introducción

Elementos a tener en  
cuenta

Términos clave

Dificultades

Ejemplos de  
problemas

Productor

```

1  item itemp;
2  while (1) {
3      produce_item (itemp);
4      while (contador==N); /* no hace nada mientras la cola est llena */
5      contador=contador+1;
6      *in = itemp;
7      if(contador==1) out=in; /* actualiza puntero de lectura de datos */
8      if(contador==N) in=NULL; /* actualiza puntero de entrada de datos */
9      else (++in) % N; /* % es el operador mdulo */
10 }

```

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

```

1  item itemc;
2  while (1) {
3      while (contador == N) continue; /* no hace nada mientras la cola est llena */

```

CLASES PARTICULARES PARA TUTORÍAS TÉCNICAS  
 LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
 ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
 CALL OR WHATSAPP. 689 45 44 70

Cartagena99

# Introducción

## Ejemplos de problemas

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

### Introducción

Elementos a tener en  
cuenta  
Términos clave  
Dificultades  
Ejemplos de  
problemas

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

**Problema 1:** coordinar lecturas y escrituras, para evitar lecturas sobre elementos no dispensados

La siguiente traza puede dar este problema:

- 1 **Productor:** produce\_item (&itemp);
- 2 **Productor:** while (contador==N);
- 3 **Productor:** contador=contador+1;
- 4 **Consumidor:** while (contador ==0);
- 5 **Consumidor:** contador = contador -1;
- 6 **Consumidor:** itemc = \*out;
- 7 **Consumidor:** if(contador ==N-1) in=out;
- 8 **Productor:** \*in = itemp;
- 9 **Productor:** if(contador==1) out=in;
- 10 **Productor:** if(

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP. 689 45 44 70

Cartagena99

# Introducción

## Ejemplos de problemas

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

### Introducción

Elementos a tener en  
cuenta

Términos clave

Dificultades

Ejemplos de  
problemas

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

**Problema 2:** Si no hay coordinación de procesos, la ejecución concurrente de **contador=contador+1** y **contador=contador-1** puede dar resultados variados, dependiendo de la traza de ejecución de instrucciones en el procesado.

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

# Introducción

## Ejemplos de problemas

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

### Introducción

Elementos a tener en  
cuenta  
Términos clave  
Dificultades  
Ejemplos de  
problemas

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

**contador=contador+1**

```
registro1 = contador
registro1 = registro1 + 1
contador = registro1
```

**contador=contador-1**

```
registro2 = contador
registro2 = registro2 - 1
contador = registro2
```

**Posible secuencia con contador=5**

```
T0(productor) : registro1 = contador(registro1 = 5)
T1(productor) : registro1 = registro1 + 1(registro1 = 6)
Cambio de contexto
T2(consumidor) : registro2 = contador(registro2 = 5)
T3(consumidor) : registro2 = registro2 - 1(registro2 = 4)
T4(consumidor) : contador = registro2(contador = 4)
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

# Memoria compartida en UNIX

## shmget

### shmget

Crea un segmento de memoria compartida o solicita acceso a un segmento de memoria existente:

```
int shmget(key_t key, int longitud, int shmflag)
```

### Salida y parámetros

- Devuelve el identificador de segmento para el programa llamante o -1 si hay error.
- **key** identifica el segmento unívocamente en la lista de segmentos compartidos mantenida por el SO.
- **longitud** es el tamaño de la región de memoria compartida.
- **shmflag** es un código octal que indica los permisos de la zona de memoria y se

CLASES PARTICULARES Y TEORIAS TÉCNICAS  
LLAMA O ENVIA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP. 689 45 44 70

Cartagena99

# Memoria compartida en UNIX

## shmat

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

shmget  
**shmat**  
Implementación  
Productor-  
Consumidor

Sincronización

Exclusión mutua

Soluciones

Cartagena99

shmat

Añade el segmento de memoria compartida a la memoria del proceso

char \*shmat(int shmid, char \*shmaddr, int shmflag)

Salida y parámetros

- Devuelve un puntero al comienzo de la zona compartida o -1 si hay error.
- **shmaddr** si es 0, el SO trata de encontrar una zona donde “mapear” el segmento compartido.
- **shmflag** es un OR de los varios elementos, de entre los que podemos destacar:

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP. 689 45 44 70

# Memoria compartida en UNIX

## Implementación Productor-Consumidor

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

shmget

shmat

Implementación  
Productor-  
Consumidor

Sincronización

Exclusión mutua

Soluciones

### Productor

```

1  #include <sys/types> #include <sys/ipc.h>
2  #include <sys/shm> #include <stdio.h>
3
4  #define SHMSZ 27
5
6  main{
7      char c, *shm, *s;
8      int shmId;
9      key_t key;
10
11     key=5678;
12     if((shmId = shmget(key,SHMSZ,IPC_CREAT|0666))
13         < 0){
14         perror(shmget);
15         exit(1);
16     }
17     if((shm = shmat(shmId,NULL,0))== (char *) -1){
18         perror(shmat);
19     }
20     while(1){
21         c = 'A';
22         while(c < 'Z'){
23             *shm = c;
24             shm++;
25             c++;
26         }
27     }
28 }
```

### Consumidor

```

1  #include <sys/types> #include <sys/ipc.h>
2  #include <sys/shm> #include <stdio.h>
3
4  #define SHMSZ 27
5
6  main(){
7      int shmId;
8      key_t key;
9      char *shm, *s;
10
11     key=5678;
12     if((shmId = shmget(key,SHMSZ,0666)) < 0){
13         perror(shmget);
14         exit(1);
15     }
16     if((shm = shmat(shmId,NULL,0))== (char *) -1){
17         perror(shmat);
18     }
19     while(1){
20         while(*shm != '\0'){
21             printf("%c", *shm);
22             shm++;
23         }
24         shm = shmat(shmId,NULL,0);
25     }
26 }
```

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
 LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
 ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
 CALL OR WHATSAPP. 689 45 44 70

Cartagena99

# Sincronización

## Labores del SO

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Labores del SO  
Interacción  
Competencia  
Cooperación

Exclusión mutua

Soluciones

① Seguir la pista de los distintos procesos activos.

② Asignar y retirar los recursos:

- Tiempo de procesador.
- Memoria.
- Archivos.
- Dispositivos de E/S.

③ Proteger los datos y los recursos físicos.

④ Los resultados de un proceso deben ser independientes de la velocidad

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

Cartagena99



Existen tres formas diferentes de interacción entre procesos:

- 1 Los procesos no tienen conocimiento de los demás.
- 2 Los procesos tienen un conocimiento indirecto de los otros.
- 3 Los procesos tienen un conocimiento directo de los otros.

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

Cuando varios procesos entran en competencia se pueden producir las siguientes situaciones:

- **Exclusión mutua:**
  - Secciones críticas:
    - Sólo un programa puede acceder a su sección crítica en un momento dado.
    - Por ejemplo, sólo se permite que un proceso envíe una orden a la impresora en un momento dado.
  - **Interbloqueo.**
  - **Inanición.**

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

## Por compartición

- Para que los procesos puedan compartir recursos adecuadamente las operaciones de escritura deben ser mutuamente excluyentes.
- La existencia de secciones críticas garantizan la integridad de los datos.

## Por cooperación

La cooperación se puede realizar por paso de mensajes. En esta situación

- No es necesario el control de la exclusión mutua.
- Puede producirse un interbloqueo:
  - Cada proceso puede estar esperando una comunicación del otro.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP. 689 45 44 70

Cartagena99

# Exclusión mutua

## Requisitos

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Requisitos

Soluciones para  
garantizarla

Soluciones  
software

- 1 Sólo un proceso debe tener permiso para entrar en la sección crítica por un recurso en un instante dado.
- 2 No puede permitirse el interbloqueo o la inanición.
- 3 Cuando ningún proceso está en su sección crítica, cualquier proceso que solicite entrar en la suya debe poder hacerlo sin dilación.
- 4 No se deben hacer suposiciones sobre la velocidad relativa de los procesos o el número de procesadores.

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

Monitores

www.cartagena99.com no se hace responsable de la información contenida en el presente documento.  
Incluido en la Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico, de 15 de mayo de 2007.  
Si la información contenida en el documento es ilícita o lesiona bienes o derechos de un tercero, haga

# Exclusión mutua

## Soluciones para garantizarla

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Requisitos

Soluciones para  
garantizarla

Soluciones  
software

- 1 Software con **Espera Activa**
- 2 Hardware
  - Deshabilitar interrupciones
  - Instrucciones especiales de hardware
- 3 Con Soporte del SO o del lenguaje de programación (biblioteca):
  - Semáforos

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

# Soluciones software

## Primer intento

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Primer intento  
Segundo intento  
Tercer intento

### Exclusión mediante el uso de turnos (corrutinas):

- Un proceso está siempre en espera hasta que obtiene permiso (turno) para entrar en su sección crítica.

### Proceso i

```
1  int turno; /* con valores de 1 a N, siendo N el numero de procesos concurrentes */
2
3  while (1) {
4      while (turno!=i);
5      --- SECCION CRTICA ---
6      turno= (i+1) %N;
7      --- RESTO DEL PROCESO ---
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

# Soluciones software

## Segundo intento

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Primer intento  
Segundo intento  
Tercer intento

- Cada proceso puede examinar el estado del otro pero no lo puede alterar.
- Cuando un proceso desea entrar en su sección crítica comprueba en primer lugar el otro proceso.
- Si no hay otro proceso en su sección crítica fija su estado para la sección crítica.

### Compartido

```
1 #define FALSE 0
2 #define TRUE 1
3 #define N 2 /* Nmero de procesos */
4 int interesado[N]; /* Todos los elementos iniciados a FALSE */
```

### Proceso i

```
1 while (interesado[j]== TRUE);
2 interesado[i]=TRUE;
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP. 689 45 44 70

# Soluciones software

## Tercer intento

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Primer intento  
Segundo intento  
Tercer intento

- Dar valor a la señal para entrar en la sección crítica antes de comprobar otros procesos.
- Si hay otro proceso en la sección crítica cuando se ha dado valor a la señal, el proceso queda bloqueado hasta que el otro proceso abandona la sección crítica.

### Compartido

```
1 #define FALSE 0
2 #define TRUE 1
3 #define N 2 /* Nmero de procesos */
4 int interesado[N]; /* Todos los elementos iniciados a FALSE */
```

### Proceso i

```
1 interesado[i]=TRUE;
2 while (interesado[i] == TRUE);
```

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP. 689 45 44 70

Cartagena99



# Soluciones software

## Cuarto intento

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Primer intento  
Segundo intento  
Tercer intento

- Un proceso activa su señal para indicar que desea entrar en la sección crítica, pero debe estar listo para desactivar la variable señal.
- Se comprueban los otros procesos. Si están en la sección crítica, la señal se desactiva y luego se vuelve a activar para indicar que desea entrar en la sección crítica. Esto se repite hasta que el proceso puede entrar en la sección crítica.

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

# Soluciones software

## Cuarto intento

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Primer intento  
Segundo intento  
Tercer intento

### Compartido

```
1 #define FALSE 0
2 #define TRUE 1
3 #define N 2 /* Nmero de procesos */
4 int interesado[N]; /* Todos los elementos iniciados a FALSE */
```

### Proceso i

```
1 interesado[i] =TRUE;
2 while (interesado[j] == TRUE){
3     interesado[i] =FALSE;
4     ---- ESPERA ----
5     interesado[i]=TRUE;
6 }
7 ---- SECCION CRITICA ----
```

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP. 689 45 44 70

Cartagena99

# Soluciones software

## Algoritmo de Dekker

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Primer intento  
Segundo intento  
Tercer intento

- Se impone un orden de actividad de los procesos.
- Si un proceso desea entrar en la sección crítica, debe activar su señal y puede que tenga que esperar a que llegue su turno.

### Compartido

```
1 #define FALSE 0
2 #define TRUE 1
3 #define N 2 /* Nmero de procesos */
4 int turno=1; /* con valores de 0 1 */
5 int interesado[N]; /* inicializado a 0 para todos los elementos del array */
```

### Proceso i

```
1 while (1) {
2     interesado[i] =TRUE;
3     while (interesado [j] ==TRUE)
4         if (turno == j) {
5             interesado[i] =FALSE;
6             while (1) {
```

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP. 689 45 44 70

Cartagena99

# Soluciones software

## Solución de Peterson

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Primer intento  
Segundo intento  
Tercer intento

### Compartido

```
1 #define FALSE 0
2 #define TRUE 1
3 #define N 2 /* Nmero de procesos */
4 int turno; /* con valores de 0 1 */
5 int interesado[N]; /* inicializado a 0 para todos los elementos del array */
```

### Proceso i

```
1 while (1) {
2     interesado[i] =TRUE;
3     turno = j; /* cambia turno al otro proceso */
4     while ((turno==j) && (interesado [j] ==TRUE));
5     --- SECCION CRITICA ---
6 }
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

# Soluciones software

## Solución de Peterson

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Primer intento  
Segundo intento  
Tercer intento

### Desglosado

#### Proceso 1

```
1 while (1) {
2     interesado[0] =TRUE;
3     turno = 1; /* cambia turno al otro proceso */
4     while ((turno==1) && (interesado [1] ==TRUE));
5     ---- SECCION CRTICA ----
6     interesado[0] =FALSE;
7     ---- RESTO DEL PROCESO ----
8 }
```

#### Proceso 2

```
1 while (1) {
2     interesado[1] =TRUE;
3     turno = 0;
```

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP. 689 45 44 70

Cartagena99

# Soluciones software

## Algoritmo de la panadería

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Primer intento  
Segundo intento  
Tercer intento

### Compartido

```
1 #define FALSE 0
2 #define TRUE 1
3 #define N n /* Nmero de procesos concurrentes */
4 int eligiendo [N]; /* con valores de 0 a n-1 */
5 int numero[N]; /* inicializado a 0 para todos los elementos del array */
```

### Proceso i

```
1 while (1) {
2     eligiendo[i] =TRUE;
3     numero[i]=max(numero[0],..., numero[n-1]) + 1;
4     eligiendo[i]=FALSE;
5     for(j=0;j<n;++j) {
6         while (eligiendo[j] ==TRUE);
7         while ( (numero[j] !=0) && ((numero[j] < numero[i])
```

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

Cartagena99

# Soluciones hardware

## Inhabilitación de interrupciones

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware

- Un proceso continuará ejecutándose hasta que solicite un servicio del sistema operativo o hasta que sea interrumpido.
- Para garantizar la exclusión mutua es suficiente con impedir que un proceso sea interrumpido.
- Se limita la capacidad del procesador para intercalar programas.
- Multiprocesador:
  - Inhabilitar las interrupciones de un procesador no garantiza la exclusión mutua.

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

# Soluciones hardware

## Instrucciones de máquina especiales

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware

### Instrucciones especiales de máquina

- Se realizan en un único ciclo de instrucción.
- No están sujetas a injerencias por parte de otras instrucciones.
- Leer y escribir.
- Leer y examinar.

### Instrucción TEST&SET

```
1  booleano TS (int i){  
2      if (i == 0) {  
3          : 1;
```

### Instrucción intercambiar

```
1  void intercambiar(int registro,  
2      int memoria)  
3  {
```

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

Cartagena99



# Soluciones hardware

## Instrucciones de máquina especiales

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware

### Ventajas

- Es aplicable a cualquier número de procesos en sistemas con memoria compartida, tanto de monoprocesador como de multiprocesador.
- Es simple y fácil de verificar.
- Puede usarse para disponer de varias secciones críticas.

### Desventajas

- Interbloqueo: si un proceso con baja prioridad entra en su sección crítica y existe otro proceso con mayor prioridad, entonces el proceso cuya prioridad es mayor obtendrá el procesador para esperar a poder entrar en la sección crítica.

Cartagena99

CLASES PARTICULARES TELEFÓNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

# Soluciones hardware

## Instrucciones de máquina especiales

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware

Con el fin de solucionar estos problemas en la sincronización se crean múltiples soluciones que veremos en las siguientes secciones y entre las que cabe destacar:

- 1 Semáforos
- 2 Mensajes
- 3 Monitores

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

Cartagena99

# Semáforos

## Definición y Propiedades

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware

- Los procesos se pueden coordinar mediante el traspaso de señales.
- La señalización se tramita mediante variable especial llamada semáforo.
- Una señal se transmite mediante una operación atómica *up* (signal).
- Una señal se recibe mediante una operación atómica *down* (wait).
- Un proceso en espera de recibir una señal es bloqueado hasta que tenga lugar la transmisión de la señal.
- Los procesos en espera se organizan en una cola de procesos.
- Dependiendo de la política de ordenamiento de procesos en espera:

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

- Un semáforo se puede ver como una variable que tiene un valor entero:
  - Puede iniciarse con un valor no negativo.
  - La operación *down* (wait) disminuye el valor del semáforo.
  - La operación *up* (signal) incrementa el valor del semáforo.
- Si la variable sólo puede tomar valores 0 y 1 el semáforo se denomina binario.
- **DOWN** (wait):
  - Comprueba el valor del semáforo:
    - Si semáforo  $> 0$ : decrementa el semáforo.
    - Si semáforo  $\equiv 0$ , el proceso se echa a dormir hasta que pueda decrementarlo.
  - **UP** (signal):

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

Cartagena99

# Semáforos

## Exclusión mutua

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware

Los semáforos garantizan la exclusión mutua en el acceso a secciones críticas.

### Compartido

```
1 #define FALSE 0
2 #define TRUE 1
3 #define N /* Nmero de procesos */
4 typedef int semaforo;
5 semaforo mutex=1; /* control de accesos a regin critica */
```

### Proceso i

```
1 while (1) {
2     down(mutex);
3     ----- SECCION CRITICA
```

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP. 689 45 44 70

Cartagena99

# Semáforos

## Productor-Consumidor

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware

### Compartido

```
1 #define N 100
2 typedef type item;
3 item array[N]; /* array circular */
4 item *in=array; /* puntero a la siguiente posicin libre */
5 item *out=NULL; /* puntero primer elemento ocupado */
6
7 typedef int semaforo;
8
9 semaforo mutex=1; /* control de accesos a regin crtica */
10 semaforo vacio=N; /* cuenta entradas vacias en el almacn se
```

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

Cartagena99

# Semáforos

## Productor-Consumidor

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware

### Solución correcta

#### Productor

```
1  item itemp;
2  while (1) {
3      produce_item (itemp);
4      down(vacio); /* decrementa entradas vacias */
5      down (mutex); /* entra en la region critica */
6      *in = itemp; /* introduce el elemento en el almacen */
7      up (mutex); /* sale de la region critica */
8      up (lleno); /* incrementa entradas ocupadas */
9  }
```

#### Consumidor

```
1  item itemc;
2  while (1) {
3      down(lleno); /* decrementa entradas ocupadas */
4      down(mutex); /* entra en la region critica */
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP. 689 45 44 70

# Semáforos

## Productor-Consumidor

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware

¿Qué hay erróneo en esta solución?

**Productor**

```
1  item itemp;
2  while (1) {
3      produce_item (itemp);
4      down (mutex); /* entra en la region critica */
5      down(vacio); /* decrementa entradas vacias */
6      *in = itemp; /* introduce el elemento en el almacen */
7      up (lleno); /* incrementa entradas ocupadas */
8      up (mutex); /* sale de la region critica */
9  }
```

**Consumidor**

```
1  item itemc;
2  while (1) {
3      down(mutex); /* entra en la region critica */
4      down(lleno); /* decrementa entradas ocupadas */
```

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP. 689 45 44 70

Cartagena99



# Semáforos

## Productor-Consumidor

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware

¿Qué hay erróneo en esta solución?

**Productor**

```
1  item itemp;
2  while (1) {
3      produce_item (itemp);
4      down(vacio); /* decrementa entradas vacias */
5      down(mutex); /* entra en la region critica */
6      *in = itemp; /* introduce el elemento en el almacen */
7      up (lleno); /* incrementa entradas ocupadas */
8      up (mutex); /* sale de la region critica */
9  }
```

**Consumidor**

```
1  item itemc;
2  while (1) {
3      down(lleno); /* decrementa entradas ocupadas */
4      down(mutex); /* entra en la region critica */
```

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP. 689 45 44 70

Cartagena99

# Semáforos

## Problema del baile de salón

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware

En el hotel Hastor de New York existe una sala de baile en la que los hombres se ponen en una fila y las mujeres en otra de tal forma que salen a bailar por parejas en el orden en el que están en la fila. Por supuesto ni un hombre ni una mujer pueden salir a bailar solos ni quedarse en la pista solos. Sin embargo no tienen por qué salir con la pareja con la que entraron.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

Cartagena99

# Semáforos

## Problema del baile de salón

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware

### Rendezvous

```

1  semaf mutex1 = 0, mutex2 = 0
2
3  Lider(mutex1,mutex2)
4  {
5      up(mutex1);
6      down(mutex2);
7  }
8
9  Seguidor(mutex1,mutex2)
10 {

```

### Solución

```

1  semaf mutex1 = 0, mutex2 = 0
2  semaf mutex3 = 0, mutex4 = 0
3
4  Hombre()
5  {
6      Lider(mutex1,mutex2);
7      Baila();
8      Seguidor(mutex3,mutex4);
9  }
10
11 Mujer()

```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
 LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
 ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
 CALL OR WHATSAPP. 689 45 44 70

# Semáforos

## Problema de los Babuinos

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware

En el parque nacional Kruger en Sudáfrica hay un cañón muy profundo con una simple cuerda para cruzarlo. Los babuinos necesitan cruzar ese cañón constantemente en ambas direcciones gracias a una cuerda. Sin embargo:

- Como los babuinos son muy agresivos, si dos de ellos se encuentran en cualquier punto de la cuerda yendo en direcciones opuestas, estos se pelearán y terminarán cayendo por el cañón y muriendo.
- La cuerda no es muy resistente y aguanta a un máximo de cinco babuinos simultáneamente. Si en cualquier instante hay más de cinco babuinos en la cuerda...

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

Cartagena99

# Semáforos

## Problema de los Babuinos

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

### Light Switch

Estructura de semáforos que permite controlar el acceso a un determinado recurso a procesos de un sólo tipo.

#### Apropiación del recurso

```
1  semaf mutex = 1 /* mutex */
2  semaf recurso = 1 /* recurso */
3  int cuentaRec = 0 /* cuenta */
4
5  lightSwitchOn(mutex,recurso,cuentaRec)
6  {
7      down (mutex);
```

#### Liberación del recurso

```
1  lightSwitchOff(mutex,recurso,cuentaRec)
2  {
3      down (mutex);
4      -- cuentaRec;
5      up (cuentaRec);
```

Cartagena99

CLASES PARTICULARES Y TEORIAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP. 689 45 44 70

# Semáforos

## Problema de los Babuinos

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware

### Solución

#### Babuino genérico

```
1 Babuino(bab,mut,rec,cRec)
2 {
3     down(bab);
4     lightSwitchOn(mut,rec,cRec);
5     CruzaBabuino();
6     lightSwitchOff(mut,rec,cRec);
7     up(babuino);
```

#### Babuinos

```
1 semaf mutVa = 1, mutVi = 1
2 semaf babVa = 5, babVi = 5
3 sem rec = 1
4 int cVa = 0, cVi = 0
5
6 BabuinoVa()
7 {
8     Babuino(babVa,mutVa,rec,cVa);
9 }
10
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP. 689 45 44 70

# Semáforos

## Problema del Barbero Dormilón

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware



Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP. 689 45 44 70

# Semáforos

## Problema del Barbero Dormilón

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware

### Aspectos a tener en cuenta

#### Control de acceso a los recursos compartidos

- Sala de espera
- Sofá
- Sillas

#### Sincronización de acciones entre cliente, barbero y cajero

- El **barbero** espera al cliente en la silla de barbero → el **cliente** se sienta
- El **cliente** espera el corte → el **barbero** indica que ha terminado
- El **barbero** espera que cliente se levante → el **cliente** indica que se ha

CLASES PARTICULARES TUTORIAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP. 689 45 44 70

Cartagena99



# Semáforos

## Problema del Barbero Dormilón

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware

### Semáforos

#### Capacidad de la tienda: max\_capacidad

- Si entra cliente, se decrementa
- Si sale cliente, se incrementa
- Si la tienda está llena, espera

#### Capacidad del sofá: sofa

- Si se sienta un cliente, se decrementa
- Si se levanta un cliente, se incrementa

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

# Semáforos

## Problema del Barbero Dormilón

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware

### Semáforos (continuación)

#### Capacidad de sillas: silla\_barbero

- Si se sienta cliente, se decrementa
- Si se levanta cliente, se incrementa
- Si las sillas están llenas, el cliente espera en los sofás

#### Ciente en la silla: cliente\_listo

- Si el barbero está dormido lo despierta
- Si no, impide que se duerma al acabar con otro cliente

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

### Semáforos (continuación)

#### Silla de barbero libre: dejar\_silla\_b

- El cliente indica que ha dejado la silla
- Otro cliente puede dejar su sofá y ocupar la silla

#### Aviso de pago: pago

- El cliente indica que paga
- El cajero despierta y cobra

#### Aviso de pago terminado: recibo

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

Cartagena99

# Semáforos

## Problema del Barbero Dormilón

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware

### Compartido

```
1  typedef int semaforo;
2  int contador = 0; /* nmero de clientes en la tienda */
3  semaforo max_capacidad=20; /* capacidad del local */
4  semaforo sofa=4; /* sofa de espera */
5  semaforo silla_barbero=3; /* sillas de la barberia */
6  semaforo cobrando=1; /* Limita a 1 el acceso a la caja */
7  semaforo mutex1=1; /* controla el acceso a contador */
8  semaforo mutex2=1; /* controla el acceso al identificador
9      de clientes */
10 semaforo cliente_listo=0; /* clientes en espera de servicio */
11 semaforo dejar_silla_b=0; /* evita colisiones en la caja, espera a
12     que se levante el cliente antes de
13     cobrarle */
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP. 689 45 44 70

# Semáforos

## Problema del Barbero Dormilón

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware

### Barabero

```

1 void barbero(void)
2 {
3     int cliente_b;
4     while (1) {
5         down (cliente_listo);
6         down (mutex2);
7         sacar_cola1(cliente_b);
8         cortar_pelo();
9         up (mutex2);
10        up (terminado[cliente_b]);
11        down (dejar_silla_b);
12        up (silla_barbero);
13        down (pago);
14        down (cobrando);
15        aceptar_pago();
    
```

### Cliente

```

1 void cliente(void)
2 {
3     int num_cliente;
4     down (max_capacidad);
5     entrar_tienda();
6     down (mutex1);
7     contador++;
8     num_cliente=contador;
9     up (mutex1);
10    down (sofa);
11    sentarse_sofa();
12    down (silla_barbero);
13    levantarse_sofa();
14    up (sofa);
15    sentarse_silla_barbero();
16    down (mutex2);
17    poner_cola1(num_cliente);
18    up (cliente_listo);
19    up (mutex2);
20    Acupar_barberado(num_cliente);
    
```

CLASES PARTICULARES TUTORÍAS TÉCNICAS  
 LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
 ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
 CALL OR WHATSAPP. 689 45 44 70

Cartagena99

# Semáforos

## Lectores–Escritores

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware

### Problema

Acceso a recursos compartidos de lectura/escritura

### Propiedades

- Cualquier número de lectores puede leer un archivo simultáneamente.
- Sólo puede escribir en el archivo un escritor en cada instante.
- Si un escritor está accediendo al archivo, ningún lector puede leerlo.

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

# Semáforos

## Lectores–Escritores

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware

### Prioridad a los lectores Compartido

```
1 typedef int semaforo;  
2 int contlect = 0; /* contador de lectores */  
3 semaforo mutex=1; /* controla el acceso a contlec */  
4 semaforo esem=1; /* controla el acceso de escritura */
```

#### Lector

```
1 void lector(void)  
2 {  
3     while (1) {  
4         down (mutex);  
5         contlect = contlect + 1;  
6         if(contlect ==1) down (esem);  
7         up (mutex);  
8         lee_recurso();  
9         down (mutex);  
10    }
```

#### Escritor

```
1 void escritor(void)  
2 {  
3     while (1) {  
4         down (esem);  
5         escribe_en_recurso();  
6         up (esem);  
7     }
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

# Semáforos

## Lectores–Escritores

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware

### Prioridad a los escritores

#### Compartido

```
1  typedef int semaforo;  
2  int contlect = 0; /* contador de lectores */  
3  int contesc = 0; /* contador de escritores */  
4  semaforo mutex1=1; /* controla el acceso a contlec */  
5  semaforo mutex2=1; /* controla el acceso a contesc */  
6  semaforo mutex3=1; /* controla el acceso al semaforo lsem */  
7  semaforo esem=1; /* controla el acceso de escritura */  
8  semaforo lsem=1; /* controla el acceso de lectura */
```

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP. 689 45 44 70

Cartagena99



# Semáforos

## Lectores–Escritores

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware

### Prioridad a los escritores

#### Lector

```
1 void lector(void)
2 {
3     while (1) {
4         down (mutex3);
5         down (lsem);
6         down (mutex1);
7         contlect = contlect + 1;
8         if(contlect ==1) down (esem);
9         up (mutex1);
10        up (lsem);
11        up (mutex3);
12        lee_recurso();
13        down (mutex1);
```

#### Escritor

```
1 void escritura(void)
2 {
3     while (1) {
4         down (mutex2);
5         contesc = contesc + 1;
6         if(contesc ==1) down (lsem);
7         up (mutex2);
8         down(esem);
9         escribe_en_recurso();
10        up (esem);
11        down (mutex2);
```

CLASES PARTICULARES Y TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP. 689 45 44 70

Cartagena99

## semget

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
int semget(int key, const void nsems, size_t semflg);
```

**Accede o crea los semáforos identificados por key**

## Significado de los parámetros

- **key**: Puede ser IPC\_PRIVATE o un identificador.
- **nsems**: Número de semáforos que se definen en el array de semáforos.
- **semflg**: Es un código octal que indica los permisos de acceso a los semáforos y a los mensajes.

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

## semop

```
#include <sys/types.h>
```

```
#include <sys/ipc.h>
```

```
#include <sys/sem.h>
```

```
int semop(int semid, struct sembuf *sops, unsigned nsops);
```

**Realiza distintas operaciones sobre el array de semáforos**

## Significado de los parámetros

- **semid**: Identificador del conjunto de semáforos.
- **sops**: Conjunto de operaciones.

Cartagena99

CLAMA O ENVIA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE ST  
CALL OR WHATSAPP. 689 45 44 70

# Monitores

## Definición

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware

- Un monitor es un módulo de software.
- Características básicas:
  - Las variables de datos locales están sólo accesibles para el monitor.
  - Un proceso entra en el monitor invocando a uno de sus procedimientos.
  - Sólo un proceso se puede estar ejecutando en el monitor en un instante dado.

## Sincronización

- La sincronización se consigue mediante variables de condición accesibles sólo desde el interior del monitor.
- Funciones básicas:

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

# Monitores Esquema

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

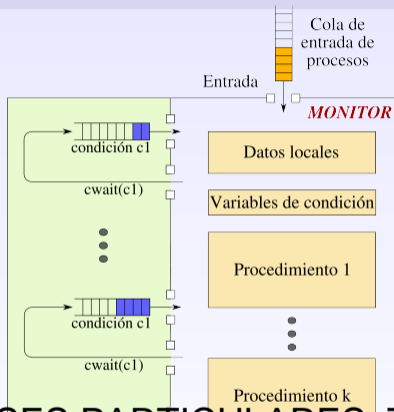
Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware



CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
 LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
 ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
 CALL OR WHATSAPP. 689 45 44 70

Cartagena99

# Mensajes

## Definición

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware

- Se utilizan como:
  - Refuerzo de la exclusión mutua, para sincronizar los procesos.
  - Medio de intercambio de información.
- La funcionalidad de paso de mensajes se implementa mediante dos primitivas:
  - send (destino, mensaje).
  - receive (origen, mensaje).

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

# Mensajes

## Sincronización

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware

- El emisor y el receptor pueden ser bloqueantes o no bloqueantes (esperando un a que se lea un mensaje o a que se escriba un nuevo mensaje).

Hay varias combinaciones posibles:

- Envío bloqueante, recepción bloqueante:
  - Tanto el emisor como el receptor se bloquean hasta que se entrega el mensaje.
  - Esta técnica se conoce como *rendezvous*.
- Envío no bloqueante, recepción bloqueante:
  - Permite que un proceso envíe uno o más mensajes a varios destinos tan rápido como sera posible.

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

# Mensajes

## Direccionamiento

Concurrencia de  
Procesos:  
Exclusión Mutua y  
Sincronización

Introducción

Memoria  
compartida en  
UNIX

Sincronización

Exclusión mutua

Soluciones  
software

Soluciones  
hardware

### Direccionamiento directo

- La primitiva `send` incluye una identificación específica del proceso de destino.
- La primitiva `receive` puede conocer de antemano de qué proceso espera un mensaje.
- La primitiva `receive` puede utilizar el parámetro `origen` para devolver un valor cuando se haya realizado la operación de recepción.

### Direccionamiento indirecto

- Los mensajes se envían a una estructura de datos compartida formada por

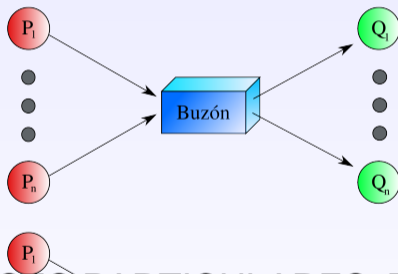
CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

Cartagena99



Procesos  
emisores

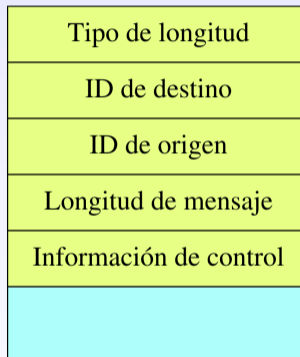
Procesos  
receptores



CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP. 689 45 44 70

Cartagena99

Cabecera



CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

Cartagena99

# Mensajes

## Lectores–Escritores

### Controlador

```

1 void controlador(void)
2 {
3     while (1) {
4         if (cont>0) {
5             if (!vacio(terminado)) {
6                 receive (terminado, msj);
7                 cont++;
8             } else if (!vacio(pedir_escritura)) {
9                 receive (pedir_escritura, msj);
10                escritor_id = msj.id;
11                cont = cont-100;
12            } else if (!vacio(pedir_lectura)) {
13                receive (pedir_lectura, msj);
14                cont--;
15                send(buzon[msj.id],"OK");
16            }
17        }
18        if (cont==0) {
19            send (buzon[escritor_id] , "OK");
20            receive (terminado, msj);

```

### Lector

```

1 void lector(int i)
2 {
3     mensaje msjl;
4     while (1) {
5         msjl=i;
6         send (pedir_lectura, msjl);
7         receive (buzon[i], msjl);
8         lee_unidad();
9         msjl=i;
10        send (terminado, msjl);
11    }
12 }

```

### Escritor

```

1 void escritura(int j)
2 {
3     mensaje msje;
4     while (1){

```

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
 LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
 ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
 CALL OR WHATSAPP. 689 45 44 70

Cartagena99

## msgsnd

```
#include <sys/msg.h>
int msgsnd(int msqid, const void *msgp, size_t msgsz, int msgflg);
```

Envía un mensaje a la cola asociada con el identificador msqid

## Significado de los parámetros

- *msgp* apunta a un buffer definido por el usuario, encabezado por un valor de tipo `long int` que indica el tipo del mensaje, seguido de los datos a enviar. Se puede implementar como una estructura:

```
struct mymsg {
```

```
    long int
```

CLASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP: 689 45 44 70

Cartagena99

## Significado de los parámetros (continuación)

- *msgsz* indica el tamaño del mensaje, que puede ser hasta el máximo permitido por el sistema.
- *msgflg* indica la acción que se debe llevar a cabo si ocurre alguno de las siguientes circunstancias:
  - El número de bytes en la cola es ya igual a *msg\_qbytes*.
  - El número total de mensajes en en colas del sistema es igual al máximo permitido.
- Si la función se ejecuta correctamente, la función devuelve un 0. En caso

Cartagena99

CLASES PARTICULARES FUTORIAS TECNICA/  
LLAMA O ENVIA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE ST  
CALL OR WHATSAPP. 689 45 44 70

## Significado de los parámetros (continuación)

- Las acciones posibles en función de *msgflg* son:
  - Si (*msgflg*&*IPC\_NOWAIT*) es distinto de 0 el mensaje no se envía y el proceso no se bloquea en el envío.
  - Si (*msgflg*&*IPC\_NOWAIT*) es 0, se bloquea la ejecución del proceso hasta que ocurre uno del estos eventos:
    - Se solventa la condición que ha provocado el bloqueo, en cuyo caso el mensaje se envía.
    - *msqid* se elimina del sistema. Esto provoca el retorno de la función con un *errno* igual a *EIDRM*.
    - El proceso que está intentando enviar el mensaje recibe una señal que debe

Cartagena99

CLAMA O ENVÍA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE ST  
CALL OR WHATSAPP. 689 45 44 70

## msgrcv

```
#include <sys/msg.h>
```

```
int msgrcv(int msqid, void *msgp, size_t msgsz, long int msgtyp, int msgflg);
```

Lee un mensaje a la cola asociada con el identificador msqid y lo guarda en el buffer apuntado por msgp

## Significado de los parámetros

- *msgp* apunta a un buffer definido por el usuario, encabezado por un valor de tipo `long int` que indica el tipo del mensaje, seguido de los datos a enviar. Se puede implementar como una estructura:

```
struct mymsg {
```

CASES PARTICULARES, TUTORÍAS TÉCNICAS  
LLAMA O ENVÍA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP. 689 45 44 70

Cartagena99

## Significado de los parámetros (continuación)

- *msgtyp* indica el tipo del mensaje a recibir. Si es 0 se recibe el primero de la cola; si es  $>0$  se recibe el primer mensaje de tipo *msgtyp*; si es  $<0$  se recibe el primer mensaje del tipo menor que el valor absoluto de *msgtyp*.
- *msgsz* indica el tamaño en bytes del mensaje a recibir. El mensaje se trunca a ese tamaño si es mayor de *msgsz* bytes y (*msgflg*&MSG\_NOERROR) es distinto de 0. La parte truncada se pierde.
- Si la función se ejecuta correctamente, la función devuelve un 0. En caso

CLASES PARTICULARES, TUTORIAS TECNICAS  
LLAMA O ENVIA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP. 689 45 44 70

Cartagena99



## Significado de los parámetros (continuación)

- *msgflg* indica la acción que se debe llevar a cabo no se encuentra un mensaje del tipo esperado en la cola. Las acciones posibles son:
  - Si (*msgflg*&IPC\_NOWAIT) es distinto de 0 el proceso vuelve inmediatamente y la función devuelve -1.
  - Si (*msgflg*&IPC\_NOWAIT) es 0, se bloquea la ejecución del proceso hasta que ocurre uno de estos eventos:
    - Se coloca un mensaje del tipo esperado en la cola.
    - *msqid* se elimina del sistema. Esto provoca el retorno de la función con un error igual a EIDRM.
    - El proceso que está intentando enviar el mensaje recibe una señal que debe

Cartagena99

CLAMA O ENVIA WHATSAPP. 689 45 44 70  
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP. 689 45 44 70