



SISTEMAS OPERATIVOS SINCRONIZACIÓN Y COMUNICACIÓN ENTRE PROCESOS

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Fernando Pérez Costoya <fperez@fi.upm.es>



- **Introducción**
- **Mecanismos de sincronización**
- **Mecanismos de comunicación local y remota**
- **Interbloqueos**

Siglas y definiciones:

PL **Proceso Ligero o *thread***

PP **Proceso Pesado**

MC **Memoria Compartida**

BD **Base de Datos**

CLASES PARTICULARES, TUTORIAS TÉCNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

proceso pesado o ligero



CONCURRENCIA

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Sucede a muchos niveles:

Concurrencia hardware

- Unidades funcionales del procesador
 - ◆ Superescalar
 - ◆ HyperThreading
- Procesador y dispositivos E/S
- Multiprocesador =
N procesadores
+ memoria compartida
 - ◆ Número de Cores
- Multicomputador =

micro

Concurrencia software

- Múltiples threads en un proceso (procesos ligeros)
- Multiprogramación de un procesador

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

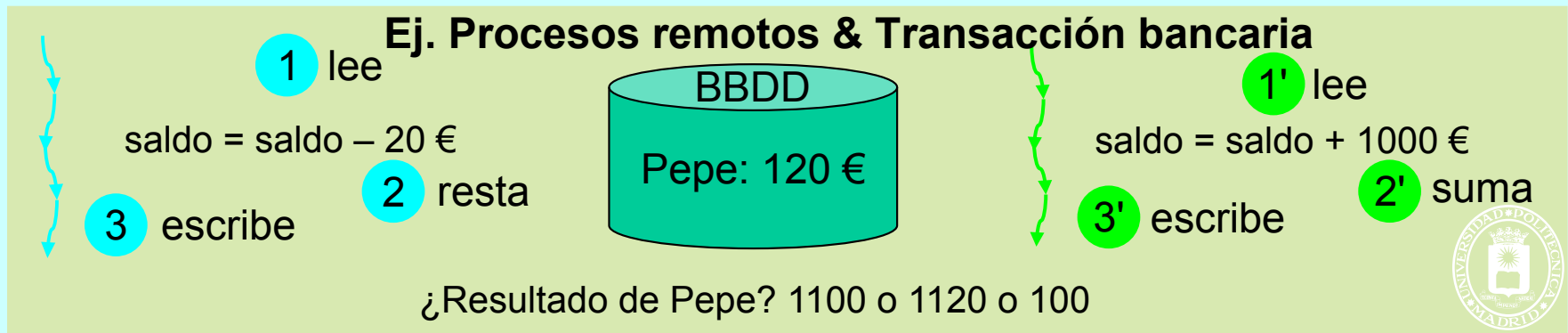
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



El acceso concurrente SIN coordinación a recursos compartidos puede NO producir los resultados esperados

Porque...

- NO se puede (ni debe) garantizar la velocidad relativa de los procesos concurrentes (== el orden en que ejecutan)



- La sentencia de alto nivel más simple requiere varias instrucciones máquina

Ej. Threads & Memoria compartida

a: 0x000A → load R1, [0x000A]

Interrupción

1 2

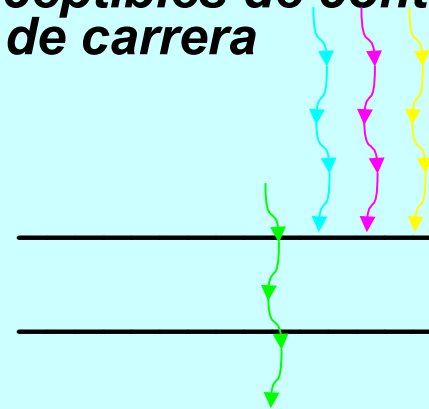
CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Cartagena99

Segmentos de código susceptibles de contener una condición de carrera

Ej. `int a; /* compartida */`
`<<Entrada a sección crítica>>`
`a = a + 1;`
`<<Salida de sección crítica>>`



- Protegeremos la sección crítica con mecanismos de sincronización que deberán garantizar:
 - Exclusión mutua: sólo entra un thread en un instante determinado. Esto implica que los threads deben esperar a que esté libre la sección crítica
 - Progreso: un thread fuera no impide que otro entre
 - Espera acotada: si espero, entraré en tiempo finito
- Los mecanismos de sincronización detienen la ejecución produciendo contención. Al ser un recurso compartido...

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



■ **Implícita:** la realiza el SO (gestor de recursos)

- ◆ Procesos independientes
- ◆ Compiten por recursos (pero lo ignoran)
- ◆ Control de acceso transparente y asociado al recurso

Ej. Procesos pesados (PPs) y... ¡¡cualquier recurso!! hw o sw

Ej. PPs y coutilización de fichero ⇒ el fichero no se corrompe pero ¡la información contenida puede no ser coherente!

■ **Explícita:** la realizan los procesos

- ◆ Procesos cooperantes
- ◆ Comparten recursos (son conscientes de ello)
- ◆ Coordinan el acceso al recurso, y para ello...
- ◆ Usan mecanismos de sincronización y/o comunicación proporcionados por: el SO, una biblioteca o el lenguaje

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Ej. Procesos distribuidos colaborando para jugar al mus en red



***Permiten resolver la sección crítica
(== coordinar el acceso a un recurso compartido)***

- **Para procesos fuertemente acoplados (== que comparten memoria)**
 - **Semáforos (adecuado para PPs con MC)**
 - **Mutex y Condiciones (adecuado para PLs)**
 - **Otros propios de lenguajes concurrentes (Monitores)**

...se suelen poner a prueba aplicados a “modelos clásicos”

- **Para procesos independientes**
 - **Semáforos, sii son servicio del SO**

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

(sin consumir ciclos de procesador)



- **Definir la información que se comparte**
 - ◆ Buffer circular
 - ◆ Registros con campos de tamaño fijo
 - ◆ Registros con campos de tamaño variable
 - ◆ ...
- **Definir el soporte de la información compartida**
 - ◆ Variables comunes (threads) . Atomicidad: load, store, test & set
 - ◆ Región de memoria . Atomicidad: load, store, test & set
 - ◆ Fichero. Atomicidad: operación E/S
- **Definir la información de control (que también se comparte) y las condiciones de acceso.**
 - ◆ Contadores, etc.
- **Seleccionar los mecanismos de control**

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



■ Productor - consumidor

- ▶ Almacenamiento compartido típico: **Buffer circular (huecos tamaño fijo)**
- ▶ Variables de control típicas:
 - N° de huecos libres
 - Punteros posición

■ Lectores y escritores

- ▶ Almacenamiento compartido típico: **Conjunto de registros**
- ▶ Granularidad del control
 - Todo el almacenamiento ⇒ Contención
 - Cada registro individual ⇒ Complejidad
- ▶ Variables de control típicas:
 - N° de lectores

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



SEMÁFORO MECANISMO DE SINCRONIZACIÓN

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



Objeto compartido con:

- Campo entero contador de valor inicial dado, que significa:

sii >0 nº de veces disponible (\approx nº plazas)

sii \leq 0 nº de threads esperando disponibilidad

- Dos acciones atómicas

```
sem_wait(s)    == down
{
  <<decrementar contador>>;
  if (<<contador>> < 0)
    <<esperar en s>>
}
```

```
sem_post(s)    == up
{
  <<incrementar contador>>
  if (<<contador>> <= 0)
```

Información de control

- Semáforo binario == máximo valor del contador es 1
- La "sección crítica" con semáforo:

```
sem_wait(binario);
<<sección crítica>>
sem_post(binario);
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



```
int sem_init (sem_t *sem, int shared, int valini);
```

```
int sem_destroy (sem_t *sem);
```

- Iniciar y destruir un semáforo SIN nombre
- PLs comparten directamente la variable “semáforo” (shared = false)
- PPs deben incluir esa variable en una zona de memoria compartida
 - ↳ Y especificar shared = true

```
sem_t *sem_open (char *name, int flag, mode_t mode, int valini);
```

```
int sem_close (sem_t *sem);
```

```
int sem_unlink (char *name);
```

- Abrir (o crear), cerrar y borrar un semáforo CON nombre
- No suelen usarse para PLs por la sobrecarga innecesaria

```
int sem_wait (sem_t *sem);
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

EQUIVALENCIAS CON ASIGNATURA DE “CONCURRENCIA”

© Latsi UPM 2016



Asignatura:	Sistemas Operativos	Concurrencia
Lenguaje prog.	C sobre POSIX	Java
Procesos ligeros	pthread	Thread
Procesos pesados	si	no
Semáforos	sem_t	java.util.concurrent.Semaphore
-----	sem_wait	.acquire
-----	sem_post	.release
Mutex	pthread_mutex_t	java.util.concurrent.locks.Lock
-----	pthread_mutex_lock	.lock
-----	pthread_mutex_unlock	.unlock

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

pthread_cond_signal .signal

pthread_cond_broadcast signalAll



MUTEX Y CONDICIONES MECANISMO DE SINCRONIZACIÓN

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



Mutual Exclusion

- == Semáforo binario **sin memoria**
 - Cerradura cerrada, llave puesta
 - Abro y cierro y me llevo la llave

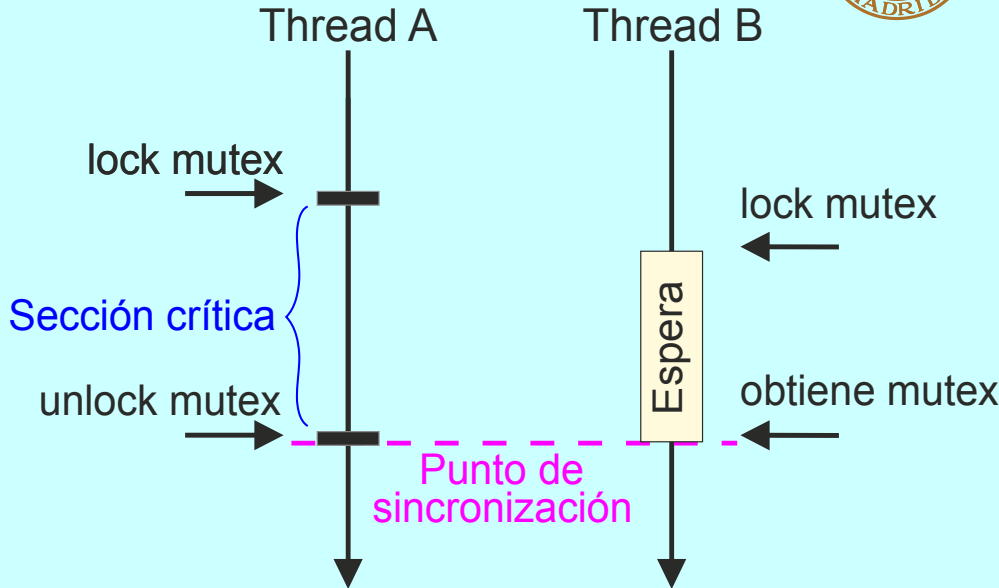
- Dos acciones atómicas

```
mutex_lock (m)
{
```

```
    if (<<no hay llave>>)
        <<esperar llave>>;
    <<abrir, entrar, cerrar y llevármela>>;
```

```
}
mutex_unlock (m)
{
```

```
    if (<<alguien espera>>)
```



- La "sección crítica" con mutex:

```
mutex_lock (mutex);
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



- Permiten liberar un mutex sin salir de la sección crítica
- Tres acciones atómicas

```
condition_wait(c,m)
{
    mutex_unlock(m);
    <<esperar aviso>>;
    mutex_lock(m);
}
condition_signal(c)
{
    if (<<alguien espera>>)
        <<avisar que siga>>;
    else (<<se pierde>>)
}
condition_broadcast(c)
{
    while (<<alguien espera>>)
```

A veces el thread que está (o posee) la sección crítica no puede continuar, porque "no se da" cierta condición que sólo podría cambiar otro thread desde dentro de la sección crítica.

- Es preciso pues:
 - Liberar **temporalmente** el mutex que protege la sección crítica...
 - ...mientras se espera a que la condición se dé.
- Todo ello:
 - Sin abandonar la sección crítica...
 - ...y de forma atómica.

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

¡Es realmente cómoda de usar!



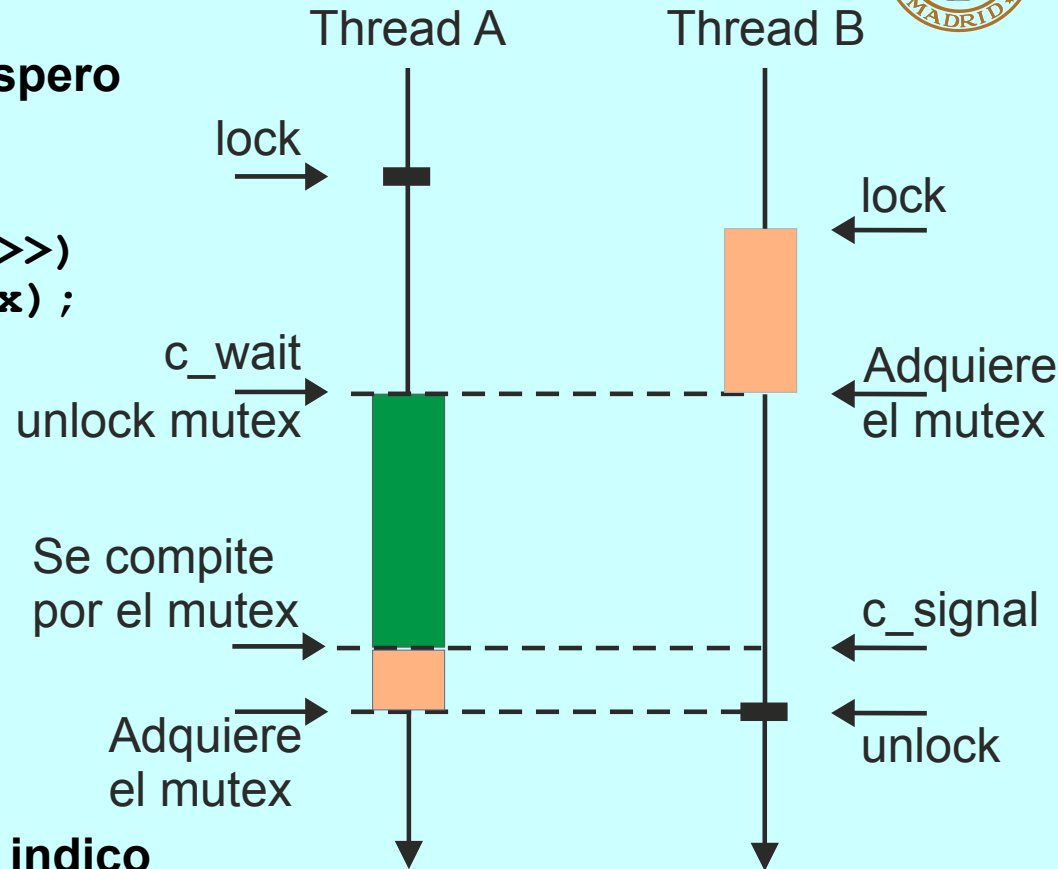
- **A: Mientras no pueda seguir, espero**

```
mutex_lock(mutex);  
<<sección crítica...>>  
while (<<no puedo continuar>>)  
    condition_wait(cond,mutex);  
<<...sección crítica>>  
mutex_unlock(mutex);
```

- **Fundamental utilizar while**

- **B: Si alguien pudiera seguir, lo indico**

```
mutex_lock(mutex);
```



Thread bloqueado

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

```
mutex_unlock(mutex);
```





Solución general

```
mutex_lock(mutex);
while (<<no puedo continuar (condición variables control)>>)
    condition_wait(cond, mutex);
<<modifica variables de control>>
mutex_unlock(mutex);
<<...sección crítica del problema>>
mutex_lock(mutex);
<<modifica variables de control>>
condition_signal(cond);
mutex_unlock(mutex);
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



```
int pthread_mutex_init (pthread_mutex_t *mutex,  
                        pthread_mutexattr_t *attr)
```

- Inicializar un mutex con atributos attr. Existen diversas llamadas para modificar los atributos, con NULL se especifican atributos por defecto
 - ▶ Atributo setpshared permite que PPs usen mutex
 - Siempre que los definan dentro de una zona de memoria compartida

```
int pthread_mutex_destroy (pthread_mutex_t *mutex);
```

- Borrar (destruir) un mutex

```
int pthread_mutex_lock (pthread_mutex_t *mutex);
```

- Competir por tomar el mutex

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



```
int pthread_cond_init (pthread_cond_t *cond,  
                      pthread_condattr_t *attr);
```

- Inicializar una variable condicional con atributos attr. Existen diversas llamadas para modificar los atributos (NULL == valores por defecto)
 - ◆ Atributo setpshared permite que PPs usen condiciones
 - Siempre que las definan dentro de una zona de memoria compartida

```
int pthread_cond_destroy (pthread_cond_t *cond);
```

- Borrar (destruir) una variable condicional

```
int pthread_cond_wait (pthread_cond_t *cond,  
                     pthread_mutex_t *mutex);
```

- Sin salir de la sección crítica, libera temporalmente el mutex que la protege, para esperar a que se "señale" la condición

```
int pthread_cond_signal (pthread_cond_t *cond);
```

- Indicar (señalar) un cambio que permitiría continuar a uno de los threads

CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Cartagena99

que esperan en la condición



```
#define BUFF_SIZE      1024
#define TOTAL_DATOS   100000
int n_datos;          /* Datos en el buffer */
int buffer[BUFF_SIZE]; /* buffer circular compartido */
pthread_mutex_t mutex; /* Acceso a sección crítica */
pthread_cond_t no_lleno, no_vacio; /* Condiciones de espera */

int main(void)
{
    pthread_t th1, th2;
    pthread_mutex_init(&mutex, NULL); /* Situación inicial */
    pthread_cond_init(&no_lleno, NULL);
    pthread_cond_init(&no_vacio, NULL);
    pthread_create(&th1, NULL, Productor, NULL); /* Arranque */
    pthread_create(&th2, NULL, Consumidor, NULL);
    pthread_join(th1, NULL); /* Esperar terminación */
    pthread_join(th2, NULL);
    pthread_mutex_destroy(&mutex); /* Destruir */
}
```

PC_PLs_mc.c

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Productor-Consumidor con mutex y condiciones (II)

© Lati UPM 2016



```
void Productor(void)
```

```
{
    int i, dato;
    for(i=0; i < TOTAL_DATOS; i++) {
        <<Producir el dato>>
        pthread_mutex_lock(&mutex);
        while(n_datos == BUFF_SIZE) //cond. espera
            pthread_cond_wait(&no_lleno, &mutex);
        buffer[i % BUFF_SIZE] = dato;
        n_datos++;
        pthread_cond_signal(&no_vacio);
        pthread_mutex_unlock(&mutex);
    }
}
```

PC_PLS_mc.c

```
void Consumidor(void)
```

```
{
    int i, dato;
    for(i=0; i < TOTAL_DATOS; i++) {
        pthread_mutex_lock(&mutex);
        while(n_datos == 0) //condición
            pthread_cond_wait(&no_vacio,
                               &mutex);
        dato = buffer[i % BUFF_SIZE];
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Lectores-Escritores con mutex y condiciones (I)

© Lati UPM 2016



```
pthread_mutex_t mutex;          /* Control de acceso */
pthread_cond_t a_leer, a_escribir; /* Condiciones de espera */
int leyendo, escribiendo;      /* Estado del acceso */
int main(void)
{
    pthread_t th1, th2, th3, th4;
    pthread_mutex_init(&mutex, NULL); /* Situación inicial */
    pthread_cond_init(&a_leer, NULL);
    pthread_cond_init(&a_escribir, NULL);
    pthread_create(&th1, NULL, Lector, NULL); /* Arranque */
    pthread_create(&th2, NULL, Escritor, NULL);
    pthread_create(&th3, NULL, Lector, NULL);
    pthread_create(&th4, NULL, Escritor, NULL);
    pthread_join(th1, NULL);          /* Esperar terminación */
    pthread_join(th2, NULL);
    pthread_join(th3, NULL);
    pthread_join(th4, NULL);
}
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

/* continúa... */


```
void Lector(void)
{
    pthread_mutex_lock(&mutex);
    while(escribiendo != 0) //condición espera
        pthread_cond_wait(&a_leer, &mutex);
    leyendo++;
    pthread_mutex_unlock(&mutex);
```

<<Lecturas simultáneas del recurso compartido>>

```
pthread_mutex_lock(&mutex);
leyendo--;
if (leyendo == 0)
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



```
void Escritor(void)
{
    pthread_mutex_lock(&mutex);
    while(leyendo !=0 || escribiendo !=0) //cond. esp
        pthread_cond_wait(&a_escribir, &mutex);
    escribiendo++;
    pthread_mutex_unlock(&mutex);

    <<Acceso en exclusiva al recurso compartido>>

    pthread_mutex_lock(&mutex);
    escribiendo--;
    pthread_cond_signal(&a_escribir);
}
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



CERROJOS

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



Servicio del SO que permite coordinar la coutilización de ficheros

- Para procesos independientes
- Permite establecer cerrojos sobre regiones de fichero

Cerrojos de tipo:

- o Compartido
 - No puede solapar con otro exclusivo
 - Sólo se puede (debe) hacer read de la región
- o Exclusivo:
 - No puede solapar con ningún otro
 - Se puede hacer read y write

Regiones relativas a:



CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
 LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
 CALL OR WHATSAPP:689 45 44 70

		Solapando con otro ya concedido de tipo:		
		Exclusivo	Compartido	Ninguno
El cerrojo pedido es de tipo:	Exclusivo	NO	NO	SI
	Compartido	NO	SI	SI

■ **Obligatorio (mandatory): afecta a procesos que no los usan (No recomendado)**

www.cartagena99.com. La información contenida en este documento es propiedad de Cartagena99. Artículo 17.1 de la Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico, de 11 de julio de 2002. Si la información contenida en el documento es ilícita o lesiona bienes o derechos de un tercero háganoslo saber y será retirada.



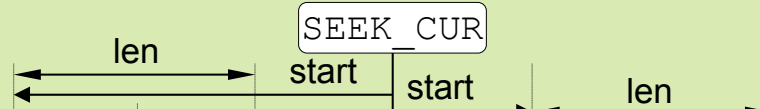
```
int fcntl(int fd, int cmd, struct flock *flockptr);
```

- **cmd:** F_GETLK, (comprueba si existe un cerrojo)
F_SETLK (establece cerrojo; no bloqueante, asíncrono)
F_SETLKW (establece cerrojo; bloqueante, síncrono)
- El cerrojo se establece sobre la región y con la modalidad indicada en:

```
struct flock {  
  short l_type;      /* F_RDLCK (compartido), F_WRLCK  
                    (exclusivo), F_UNLCK (elimina) */  
  short l_whence;   /* SEEK_SET, SEEK_CUR, SEEK_END */  
  off_t l_start;    /* Desfase relativo a l_whence */  
  off_t l_len;      /* Tamaño. 0 == hasta EOF */  
  pid_t l_pid;      /* Sólo F_GETLK, primer pid con lock*/  
}
```



fd



CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70





■ Cerrojos fcntl

- ▶ Los cerrojos se asocian al proceso
- ▶ Si el proceso cierra un descriptor cualquiera del fichero (que puede tener más de uno) se pierden los cerrojos
- ▶ Los cerrojos no se heredan (Posix)

■ A título informativo indicamos que Linux permite cerrojos obligatorios

▶ En el sistema de ficheros

- Al montar el sistema de ficheros hay que habilitarlos:

`mount -o mand`

▶ En el fichero

- Hay que deshabilitar los permisos de grupo y habilitar los permisos: `set-group-ID (SGID)`

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



Lector

L_PPs_cf.c

```
int main(void)
{
    int fd, val, cnt;
    struct flock fl;
    fl.l_whence = SEEK_SET;
    fl.l_start = 0;
    fl.l_len = 0; Cierra todo el fichero
    fl.l_pid = getpid();
    fd = open("BD", O_RDONLY);
    for (cnt = 0; cnt < 10; cnt++)
    {
        fl.l_type = F_RDLCK; Compartido
        fcntl(fd, F_SETLKW, &fl); Sinc.
        lseek(fd, 0, SEEK_SET);
        read(fd, &val, sizeof(int));
        printf("%d\n", val);
        /*Lecturas simultáneas
        del recurso compartido*/
```

Escritor

L_PPs_cf.c

```
int main(void)
{
    int fd, val, cnt;
    struct flock fl;
    fl.l_whence = SEEK_SET;
    fl.l_start = 0;
    fl.l_len = 0; Cierra todo el fichero
    fl.l_pid = getpid();
    fd = open("BD", O_RDWR);
    for (cnt = 0; cnt < 10; cnt++)
    {
        fl.l_type = F_WRLCK; Exclusivo
        fcntl(fd, F_SETLKW, &fl); Sinc.
        lseek(fd, 0, SEEK_SET);
        read(fd, &val, sizeof(int));
        val++; /*Acceso exclusivo*/
        lseek(fd, 0, SEEK_SET);
        write(fd, &val, sizeof(int));
```

Cartagena99

CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Ejemplo de cerrojos: cuentas bancarias (i)

© Lati UPM 2016



```
fperez@box1: ~/so5
#define FICH "cuentas.dat"

struct cuenta {
    char titular[32];
    float saldo;
};

int crear_cuenta(char tit[], float sal_ini) {
    int fd, pos;
    struct cuenta c;

    fd=open(FICH, O_WRONLY|O_CREAT|O_APPEND, 0600);
    strcpy(c.titular, tit);
    c.saldo = sal_ini;
    write(fd, &c, sizeof(struct cuenta));
    pos = lseek(fd, 0, SEEK_CUR);
    return pos / sizeof(struct cuenta);
}
```

27,0-1 9%

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



```
fperez@box1: ~/so5
int imprimir_cuenta(int n_cnt) {
    int fd;
    struct flock fl;
    struct cuenta c;

    if ((n_cnt * sizeof(struct cuenta)) > tam_fichero())
        return -1;
    fd=open(FICH, O_RDONLY);
    fl.l_type = F_RDLCK;
    fl.l_whence = SEEK_SET;
    fl.l_start = (n_cnt-1) * sizeof(struct cuenta);
    fl.l_len = sizeof(struct cuenta);
    fcntl(fd, F_SETLKW, &fl);
    lseek(fd, (n_cnt-1) * sizeof(struct cuenta), SEEK_SET);
    read(fd, &c, sizeof(struct cuenta));
    printf("Titular %s saldo %f\n", c.titular, c.saldo);
    fl.l_type = F_UNLCK;
    fcntl(fd, F_SETLKW, &fl);
    close(fd);
    return 0;
}
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



```
fperez@box1: ~/so5
int operacion_cuenta(int n_cnt, float val) {
    int fd;
    struct flock fl;
    struct cuenta c;

    if ((n_cnt * sizeof(struct cuenta)) > tam_fichero())
        return -1;
    fd=open(FICH, O_RDWR);
    fl.l_type = F_WRLCK;
    fl.l_whence = SEEK_SET;
    fl.l_start = (n_cnt-1) * sizeof(struct cuenta);
    fl.l_len = sizeof(struct cuenta);
    fcntl(fd, F_SETLKW, &fl);
    lseek(fd, (n_cnt-1) * sizeof(struct cuenta), SEEK_SET);
    read(fd, &c, sizeof(struct cuenta));
    c.saldo+=val;
    printf("Saldo actual %f\n", c.saldo);
    lseek(fd, (n_cnt-1) * sizeof(struct cuenta), SEEK_SET);
    write(fd, &c, sizeof(struct cuenta));
    fl.l_type = F_UNLCK;
    fcntl(fd, F_SETLKW, &fl);
    return fd;
}
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



MECANISMOS DE COMUNICACIÓN

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



- **Dirección:** Identifica al receptor o al emisor (remite)
- **Tipo**
 - Sin nombre: Se hereda el identificador del creador del mecanismo
 - Nombre textual o simbólico (/home/jfelipe/mififo, www.fi.upm.es)
 - Nombre físico (fd = 5, IP = 138.100.8.100 TCP = 80)
 - Estructura en árbol
- **Ámbito**
 - Procesos emparentados (creador y sus descendientes)
 - Local: Misma máquina
 - Remoto: Máquinas distintas
- **Servidor de nombres**
 - Convierte el nombre textual en físico

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



Al crearlo (o abrirlo)

- Según la forma de nombrarlo
 - Sin nombre (Ej. PIPE)
 - Con nombre local (Ej. FIFO)
 - Con nombre de red (Ej. *Socket*)
- Según el identificador devuelto
 - Descriptor de fichero (sii es servicio ofrecido por el SO)
 - Identificador propio (sii es ofrecido por biblioteca)

Al usarlo (a través de su ID)

- Según el flujo de datos
 - Unidireccional
 - Bidireccional
- Según capacidad de memoria
 - Sin *buffering*
 - Con *buffering*
- Según bloquee o no
 - Síncrono (bloqueante)

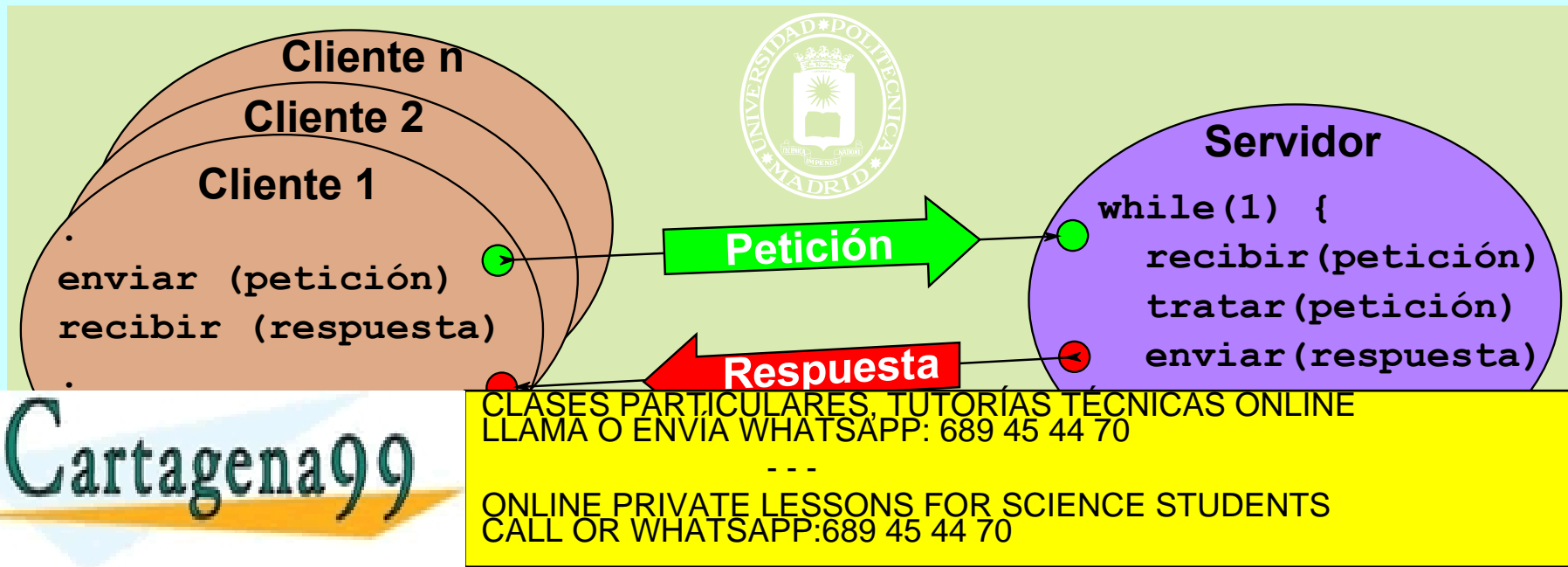
Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Modela acceso a gestor (local o remoto) de recurso

- Protocolo Petición-Respuesta para acceso a gestor de recurso
- Servidor == proveedor de servicio == gestor del recurso
- Por extensión, también se dice servidor a la máquina donde éste reside
- Cliente: Accede al proveedor siguiendo el protocolo
- Comunicación N (clientes) a 1 (servidor) o N a M





- El modelo Cliente-Servidor clásico es centralizado
- En el modelo P2P (“entre pares” o “entre iguales”):
 - Cada miembro es, a la vez, cliente y servidor
 - La información o recurso está distribuido entre los miembros
 - Red no estructurada: cada miembro tiene lo que quiere
 - Red estructurada: la información o recurso está repartido según determinados criterios
 - La gestión del directorio y del enrutamiento puede estar:
 - Centralizado (Napster y Audiogalaxy)
 - Totalmente distribuido (Ares Galaxy, Gnutella, Freenet y Kademia)
 - Híbrido (Bittorrent, eDonkey2000 y Direct Connect)

■ Características del P2P

➤ Escalable

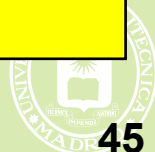
➤ Costes distribuidos

➤ Robusto

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70





MECANISMOS DE COMUNICACIÓN LOCAL

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

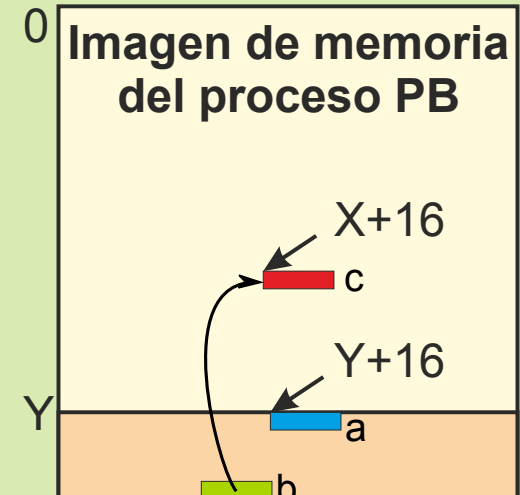
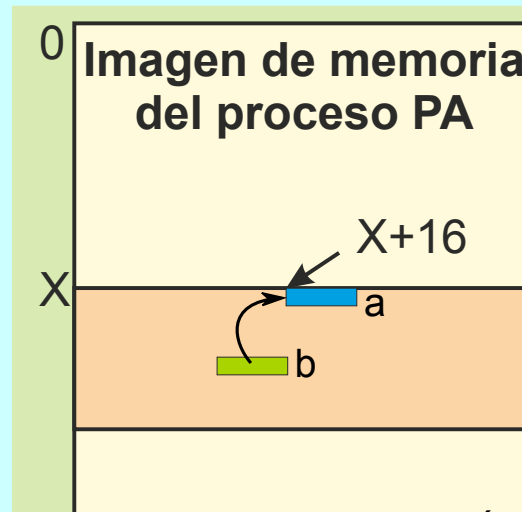
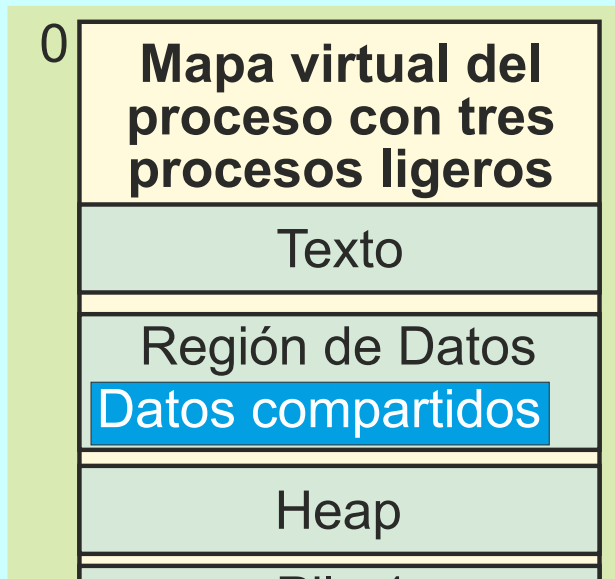


■ Procesos ligeros

- ▶ Todo su espacio de memoria está compartido:
 - Variables globales, variables estáticas, etc.
- ▶ Cada uno "usa" su propia pila

■ Procesos pesados con memoria compartida

- ▶ Regiones compartidas (mmap)
- ▶ Declaración independiente de variables proyectadas en MC



Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



Mecanismo del SO que da resuelto el esquema de comunicación "Productores-Consumidores" entre procesos pesados emparentados

- Sin nombre (sólo visible por quien lo crea y procesos derivados de éste)
- Unidireccional
- Con buffering

```
int pipe (int fds[2]);
```

- Devuelve dos descriptores de

```
read (fds[0], datos, n);
```

- Si vacío \Rightarrow se bloquea el lector
- Si con p bytes \Rightarrow
 - ▶ Si $p \geq n$ devuelve n
 - ▶ Si $p < n$ devuelve p
- Si vacío y no hay escritores \Rightarrow devuelve 0 (indicando EOF)

```
write (fds[1], datos, n);
```

- Si lleno \Rightarrow se bloquea el escritor
- Si no hay lectores \Rightarrow el escritor

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Las lecturas y escrituras pequeñas
son atómicas (no se mezclan)



Mecanismo del SO

para comunicar

procesos pesados no remotos

- Se crean como FIFO
- Se usan como fichero
- Se comportan como PIPE

```
int open (char *name,  
         int flags);
```

- Abrir para lectura, escritura o ambas
- Bloquea hasta que se abran los dos extremos (salvo O_NONBLOCK)

```
int mkfifo (char *name,  
          mode_t mode);
```

- Crea un FIFO con nombre y permisos iniciales dados

```
read (fd_in, datos, n);
```

```
write (fd_out, datos, n);
```

- Semántica PIPE

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Eliminar un FIFO



Crear FIFO



```
int main(int argc, char *argv[])
{
    if (mkfifo(argv[1], 0666) == 0)
        return 0;
    perror(argv[0]);
    return 1;
}
```

Crea el FIFO

Productor



```
int main(int argc, char *argv[])
{
    int fd;
    char ch;
    fd = creat(argv[1], 0666);
    if (fd == -1) {
        perror(argv[0]);
        return 1;
    }
}
```

Abre el FIFO para escritura

Consumidor



```
int main(int argc, char *argv[])
{
    int fd;
    char ch;
    fd = open(argv[1], O_RDONLY);
    if (fd == -1) {
        perror(argv[0]);
        return 1;
    }
}
```

Abre el FIFO para lectura

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



Ej. Ejecución de un mandato equivalente a: `ls -l | sort`
a través de un FIFO

```
$ mkfifo FIFO
$ ls -l FIFO
prw-r--r-- 1 frosal frosal    0 Apr 28 21:11 FIFO
$ sort < FIFO &
[135]
$ ls -l > FIFO
-rw-r----- 1 frosal frosal    0 Apr 29 00:08 Archivo
brw-rw---- 2 frosal disk   3,65 Sep 27  2000 Bloques
crw----- 2 root  root    4,1 Apr  8 13:02 Caracteres
drwxr-x--- 2 frosal frosal 4096 Apr 29 00:08 Directorio
lrwxrwxrwx 1 frosal frosal    7 Apr 29 00:10 Enlace -> Archivo
prw-r--r-- 1 frosal frosal    0 Apr 28 21:11 FIFO
```

Cartagena99

CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

¿Qué pasa con el FIFO del ejemplo?



MECANISMO DE COMUNICACIÓN REMOTA

SOCKET

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



Mecanismo del SO para comunicación dentro del mismo dominio

- Con nombre (dirección)
- Bidireccional
- Con buffering
- Bloqueante o no

```
int socket (int dominio,  
           int tipo, int protocolo);
```

- Crea un *socket* (sin dirección) del dominio, tipo y protocolo dados y devuelve descriptor asociado **sd**
- **Dominio == familia de direcciones**

➤ **AF_UNIX: intra-máquina**
(Dir. = nombre de fichero)

■ Tipo

➤ Stream (SOCK_STREAM)

- Orientado a flujo de datos
- CON conexión
- Fiable: asegura entrega y orden
- [≈ Conversación telefónica]

➤ Datagrama (SOCK_DGRAM)

- Orientado a mensajes
- SIN conexión
- No fiable: pérdida y desorden
- [≈ Correspondencia postal]

■ Protocolo

➤ Mensajes y reglas de intercambio entre comunicantes

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

dominio, pero diferente tipo de direcciones

– IPPROTO_UDP, para datagrama



- Hay arquitecturas de computador que representan los números enteros de manera distinta: Little-endian vs. Big-endian
- Para transmitir enteros, hay que convertirlos a/de formato independiente

htonl ()

htons ()

- Convierten enteros largos y cortos de formato del host a formato de red

ntohl ()

ntohs ()

- Convierten enteros largos y cortos de formato del red a formato del host
- Por ejemplo, el campo `sockaddr_in.sin_port` (número de puerto) debe estar en formato red

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

(8 bits)



- Las direcciones dependen del dominio, pero los servicios no (uso de cast)
 - struct sockaddr
estructura genérica de dirección
 - struct sockaddr_in
estructura específica AF_INET
 - Debe iniciarse a 0 (memset)
 - sin_family: AF_INET
 - sin_addr: dirección del host (32 bits) (4 octetos [0..255])
 - sin_port: número de puerto (16 bits) (1024 reservados)
- Una transmisión IP está caracterizada por cinco parámetros:
 - Protocolo (UDP o TCP)
 - Dirección host + puerto origen
 - Dirección host + puerto destino
- Cada socket debe estar asociado a:
 - una dirección local única
 - y una dirección remota sii:
 - está conectado
 - o para cada mensaje
- Para los usuarios son texto
"138.100.8.100" ó "laurel.datsi.fi.upm.es"

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Conversion a binario (formato red) desde dominio punto



```
int socket (int dominio, int tipo, int protocolo);
```

- Crea un socket (sin dirección) del dominio, tipo y protocolo dados

```
int bind (int sd, struct sockaddr *dir, int tam);
```

- Asociar a una dirección local

```
int connect (int sd, struct sockaddr *dir, int tam);
```

- Asociar a una dirección remota (cliente)

```
int listen (int sd, int backlog);
```

- Prepara para aceptar conexiones (servidor)

```
int accept (int sd, struct sockaddr *dir, int *tam);
```

- Aceptación de una conexión (servidor)

```
int send (int sd, char *mem, int tam, int flags);
```

```
int recv (int sd, char *mem, int tam, int flags);
```

- Transmisión para conectados (también read y write)

```
int sendto (int sd, char *mem, int tam, int flags, struct sockaddr *dir, int tam);
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

- Transmisión para NO conectados



Cliente

(Inicia la conversación)

`socket`

- Adquirir buzón UDP

`bind`

- Asignarle una dirección libre

`sendto`

- Enviar carta con remite: **¿Hola?**

`recvfrom`

- Recoger respuesta

Servidor SIN estado

(Recibe mensajes y los atiende)

`socket`

- Adquirir buzón UDP

`bind`

- Asignarle dirección bien conocida

`recvfrom`

- Recoger carta
- Tomar dirección del remitente

`sendto`

- Enviar al remitente: **¿Dígame?**

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



Máquina Cliente A

Proceso Cliente

cd=socket(UDP)

bind(cd,0)

sendto(cd,B:7)

puertos UDP

1025

Máquina Servidor B

Demonio Servidor SIN estado

sd=socket(UDP)

bind(sd,7)

recvfrom(sd,caddr)

sendto(sd,caddr)

puertos UDP

7

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

desorden, duplicados



```
int main(int argc, char * argv[])
{
    int cd, size ;
    struct hostent * hp;
    struct sockaddr_in s_ain, c_ain;
    unsigned char byte;

    hp = gethostbyname(argv[1]); /* por ejemplo, otilio.fi.upm.es */
    memset((char *)&s_ain, 0, sizeof(s_ain));
    s_ain.sin_family = AF_INET;
    memcpy (&(s_ain.sin_addr), hp->h_addr, hp->h_length);
    s_ain.sin_port = htons(7); /* echo port */

    cd = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
    memset((char *)&c_ain, 0, sizeof(c_ain));
    c_ain.sin_family = AF_INET;
    bind(cd, (struct sockaddr *)&c_ain, sizeof(c_ain));
    size = sizeof(s_ain);
    while(read( 0, &byte, 1) == 1) {
        sendto(cd, &byte, 1, 0, (struct sockaddr *)&s_ain, size);
        recvfrom(cd, &byte, 1, 0, (struct sockaddr *)&s_ain, &size);
        write(1, &byte, 1);
    }
}
```

O se hace el bind explícito o el sendto lo hará implícitamente

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



```
int main(void)
{
    int sd, size;
    unsigned char byte;
    struct sockaddr_in s_ain, c_ain;

    sd = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);

    memset((char *)&s_ain, 0, sizeof(s_ain));
    s_ain.sin_family = AF_INET;
    s_ain.sin_addr.s_addr = INADDR_ANY; /*Cualquier origen*/
    s_ain.sin_port = htons(7); /* echo server */
    bind(sd, (struct sockaddr *)&s_ain, sizeof(s_ain));
    size = sizeof(c_ain);

    while (1) {
        recvfrom(sd, &byte, 1, 0, (struct sockaddr *)&c_ain, &size);
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



Cliente (Inicia la conversación)

socket

- Adquirir teléfono

bind

- Contratar línea (nº de teléfono)

connect

- Descolgar
- Marcar

-
- Esperar establecimiento llamada

Centralita Servidor (Recibe llamada y redirige)

socket

- Adquirir centralita

bind

- Contratar línea (nº de teléfono)

listen

- Dimensionar centralita

accept

-
- Esperar establecimiento llamada
 - Redirigir a teléfono de servicio

Servidor dedicado (Atiende la conversación)

send & recv

- Saludo:

send & recv

¿Hola?

send & recv

- Saludo:

send & recv

¿Qué desea?

CLASES PARTICULARES, TUTORIAS TÉCNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Cartagena99

CLIENTE-SERVIDOR TCP (SERVIDOR SECUENCIAL)

© Latsi UPM 2016



Máquina Cliente A

Proceso Cliente

cd=socket(TCP)

bind(cd,1025)

connect(cd,B:7)

send(cd)

recv(cd)

puertos TCP

puertos TCP

Máquina Servidora B

Demonio Servidor

sd=socket(TCP)

bind(sd,7)

listen(sd,5)

cd=accept(sd)

recv(cd)

send(cd)

close(cd)

Conexión:

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

entrega y orden

Cartagena99

CLIENTE-SERVIDOR TCP (SERVIDOR CONCURRENTE FORK)

© Latsi UPM 2016



Máquina Cliente A

Proceso Cliente

cd=socket(TCP)

bind(cd,1025)

connect(cd,B:7)

send(cd)

close(cd)

puertos TCP

puertos TCP

Máquina Servidora B

Demonio Servidor

sd=socket(TCP)

bind(sd,7)

listen(sd,5)

cd=accept(sd)

close(cd)

fork()

close(sd)

recv(cd)

Servidor dedicado

Conexión:

CLASES PARTICULARES, TUTORIAS TÉCNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

entrega y orden

Cartagena99



Máquina Cliente A

Proceso Cliente

cd=socket(TCP)

bind(cd,1025)

connect(cd,B:7)

send(cd)

close(cd)

puertos TCP

puertos TCP

Máquina Servidora B

Demonio Servidor

sd=socket(TCP)

bind(sd,7)

listen(sd,5)

cd=accept(sd)

pthread_create()

Thread main

Thread servicio

recv(cd)

Conexión:

CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

entrega y orden

Cartagena99



T C P _ c . c

```
int main(int argc, char * argv[])
{
    int cd;
    struct hostent * hp;
    struct sockaddr_in s_ain;
    unsigned char byte;

    hp = gethostbyname(argv[1]);
    memset((char *)&s_ain, 0, sizeof(s_ain));
    s_ain.sin_family = AF_INET;
    memcpy (&(s_ain.sin_addr), hp->h_addr, hp->h_length); /* IP */
    s_ain.sin_port = htons(7); /* echo port */

    cd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    connect(cd, (struct sockaddr *)&s_ain, sizeof(s_ain));

    while(read( 0, &byte, 1) == 1) {
        send(cd, &byte, 1, 0); /* Bloqueante */
        recv(cd, &byte, 1, 0); /* Bloqueante */
        write(1, &byte, 1);
    }
}
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



TCP_s.c

```
int main(void)
{
    int sd, cd, size;
    unsigned char byte;
    struct sockaddr_in s_ain, c_ain;

    sd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

    memset((char *)&s_ain, 0, sizeof(s_ain));
    s_ain.sin_family = AF_INET;
    s_ain.sin_addr.s_addr = INADDR_ANY; /*Cualquier origen*/
    s_ain.sin_port = htons(7); /* echo port */

    bind(sd, (struct sockaddr *)&s_ain, sizeof(s_ain));

    listen(sd, 5); /* 5 = tamaño cola */
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



TCP_s.c

```
while(1) {
    size = sizeof(c_ain);
    cd = accept(sd, (struct sockaddr *)&c_ain, &size);
    switch(fork()) {
    case -1:
        perror("echo server");
        return 1;
    case 0:
        close(sd);
        while(recv(cd, &byte, 1, 0) == 1) /*Bloqueante*/
            send(cd, &byte, 1, 0);          /*Bloqueante*/
        close(cd);
        return 0;
    default:
        close(cd);
    } /* switch */
}
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



INTERBLOQUEOS

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

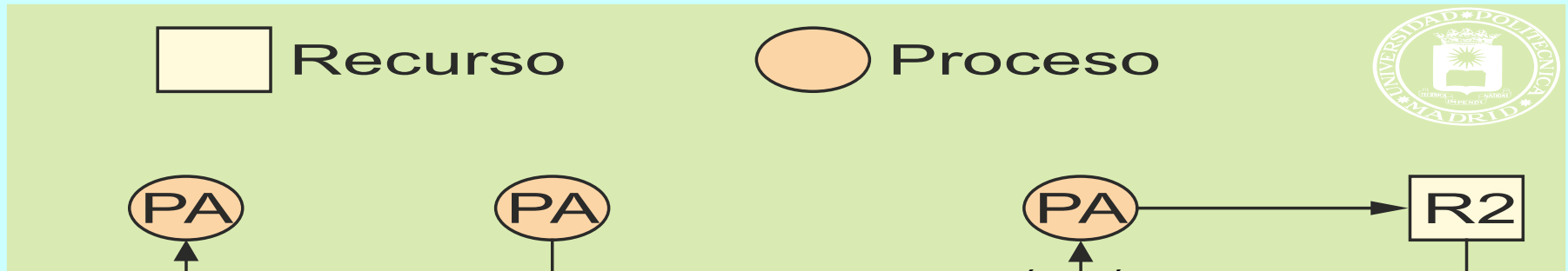
ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



Un conjunto de procesos está en interbloqueo cuando cada proceso está bloqueado en espera de un recurso (o evento) que está asignado a (o sólo puede producir) un proceso (también bloqueado) del conjunto

- En grafo de asignación de recursos se visualizan como bucles

Ej. PA tiene R1 y necesita R2
PB tiene R2 y necesita R1



Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



■ Ej. Si S y Q son semáforos binarios

P0	P1
<code>sem_wait(S)</code>	<code>sem_wait(Q)</code>
<code>sem_wait(Q)</code>	<code>sem_wait(S) ← !!</code>
...	...
<code>sem_post(S)</code>	<code>sem_post(Q)</code>
<code>sem_post(Q)</code>	<code>sem_post(S)</code>

■ Ej. PIPE y lectura

```
...
pipe(pp) ;
read(pp[0], ...) ; ← !!
/* NO termina */
/* !Quedan escritores! */
...
```

■ Condiciones para el interbloqueo

- ▶ Exclusión mutua:
Un único usuario por recurso
- ▶ Mantener y esperar:
Espero sin liberar lo que tengo

■ ¿Cómo tratar el interbloqueo?

- ▶ No hacer nada: Alg. del avestruz
- ▶ Detectarlo y notificarlo
- ▶ Prevenirlo: eliminar alguna

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Puede darse esta situación (bucle)

banquero, etc.)



```
fperez@box1: ~/so5/SC_Codigo
int lockOP(int fd, short ty, off_t st, off_t le) {
    struct flock fl = {.l_type=ty, .l_whence=SEEK_SET, .l_start=st, .l_len=le};
    return fcntl(fd, F_SETLKW, &fl); }

int transferencia_cuentas(int n_cnt_org, int n_cnt_dst, float val) {
    int fd; struct cuenta c;
    if (((n_cnt_org * sizeof(struct cuenta)) > tam_fichero()) ||
        ((n_cnt_dst * sizeof(struct cuenta)) > tam_fichero())) return -1;
    fd=open(FICH, O_RDWR);
    if (lockOP(fd, F_WRLCK, (n_cnt_org-1)*sizeof(struct cuenta), sizeof(struct cuenta))<0){
        perror("error en fcntl"); close(fd); return -1; }
    // fuerce una parada aquí (getchar) y ejecute 2 transferencias cruzadas -> interbloqueo
    if (lockOP(fd, F_WRLCK, (n_cnt_dst-1)*sizeof(struct cuenta), sizeof(struct cuenta))<0){
        perror("error en fcntl"); // SO detecta interbloqueo y devuelve un error
        lockOP(fd, F_UNLCK, (n_cnt_org-1)*sizeof(struct cuenta), sizeof(struct cuenta));
        close(fd); return -1; }
    pread(fd, &c, sizeof(struct cuenta), (n_cnt_org-1) * sizeof(struct cuenta));
    c.saldo-=val;
    printf("Cuenta origen %d Saldo resultante %f\n", n_cnt_org, c.saldo);
    pwrite(fd, &c, sizeof(struct cuenta), (n_cnt_org-1) * sizeof(struct cuenta));
    lockOP(fd, F_UNLCK, (n_cnt_org-1)*sizeof(struct cuenta), sizeof(struct cuenta));
    pread(fd, &c, sizeof(struct cuenta), (n_cnt_dst-1) * sizeof(struct cuenta));
    c.saldo+=val;
    printf("Cuenta destino %d Saldo resultante %f\n", n_cnt_dst, c.saldo);
```

Cartagena99

CLASES PARTICULARES, TUTORIAS TECNICAS ONLINE
LLAMA O ENVIA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

SO detecta el problema y retorna error en el segundo lock del segundo proceso.



```
fperez@box1: ~/so5/SC_Codigo
int lockOP(int fd, short ty, off_t st, off_t le) {
    struct flock fl = {.l_type=ty, .l_whence=SEEK_SET, .l_start=st, .l_len=le};
    return fcntl(fd, F_SETLKW, &fl); }

int transferencia_cuentas(int n_cnt_org, int n_cnt_dst, float val) {
    int fd; struct cuenta c; int cnt1=n_cnt_org, cnt2=n_cnt_dst;
    if (((n_cnt_org * sizeof(struct cuenta)) > tam_fichero()) ||
        ((n_cnt_dst * sizeof(struct cuenta)) > tam_fichero())) return -1;
    if (n_cnt_org > n_cnt_dst) {cnt1=n_cnt_dst, cnt2=n_cnt_org;}
    fd=open(FICH, O_RDWR);
    if (lockOP(fd, F_WRLCK, (cnt1-1)*sizeof(struct cuenta), sizeof(struct cuenta))<0){
        perror("error en fcntl"); close(fd); return -1; }
    // fuerce una parada aquí (getchar) y ejecute 2 transferencias cruzadas-> NO interbloqueo
    if (lockOP(fd, F_WRLCK, (cnt2-1)*sizeof(struct cuenta), sizeof(struct cuenta))<0){
        perror("error en fcntl");
        lockOP(fd, F_UNLCK, (cnt1-1)*sizeof(struct cuenta), sizeof(struct cuenta));
        close(fd); return -1; }
    pread(fd, &c, sizeof(struct cuenta), (n_cnt_org-1) * sizeof(struct cuenta));
    c.saldo-=val;
    printf("Cuenta origen %d Saldo resultante %f\n", n_cnt_org, c.saldo);
    pwrite(fd, &c, sizeof(struct cuenta), (n_cnt_org-1) * sizeof(struct cuenta));
    lockOP(fd, F_UNLCK, (n_cnt_org-1)*sizeof(struct cuenta), sizeof(struct cuenta));
    pread(fd, &c, sizeof(struct cuenta), (n_cnt_dst-1) * sizeof(struct cuenta));
    c.saldo+=val;
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

Se previene el problema eliminando una condición necesaria.

Se previene el problema eliminando una condición necesaria.



- La concurrencia sucede a muchos niveles
- Implica compartición de recursos
- Condición de carrera == No se puede garantizar la velocidad relativa de los procesos concurrentes ← pueden suceder resultados incorrectos
- Sección crítica == trozos de código susceptibles de condición de carrera
- Para coordinar el acceso usamos:
 - ◆ Mecanismos de sincronización
 - ◆ Mecanismos de comunicación
- Mecanismos de sincronización:
 - ◆ Acciones atómicas sobre base de instrucciones máquina atómicas
 - ◆ Modelos clásicos: modelan arquitecturas clave en concurrencia
- Para procesos fuertemente acoplados (== que comparten memoria):
 - ◆ Semáforos == contador
 - ◆ Mutex == cerrojo

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70



- **Mecanismos de comunicación**
== **Recurso compartido + Mecanismo de sincronización**
 - ◆ **Con o sin nombre, descriptor de fichero o identificador propio, uni o bidireccional, con o sin *buffer*, bloqueante o no**
- **Para procesos locales:**
 - ◆ **PIPE == mecanismo para N productores y M consumidores**
 - ◆ **FIFO == crear con `mkfifo`, usar como fichero, semántica de PIPE**
- **Para procesos remotos:**
 - ◆ **Sockets: dominio, tipo, protocolo**
 - **Datagrama == correspondencia postal**
 - ***Stream* == conversación telefónica**
 - ◆ **Servicios genéricos PERO direcciones específicas del dominio**
 - ◆ **Formato de red**

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70