# Lectores/Escritores

Juan Quintela – Javier París
{quintela,javier.paris}@udc.es

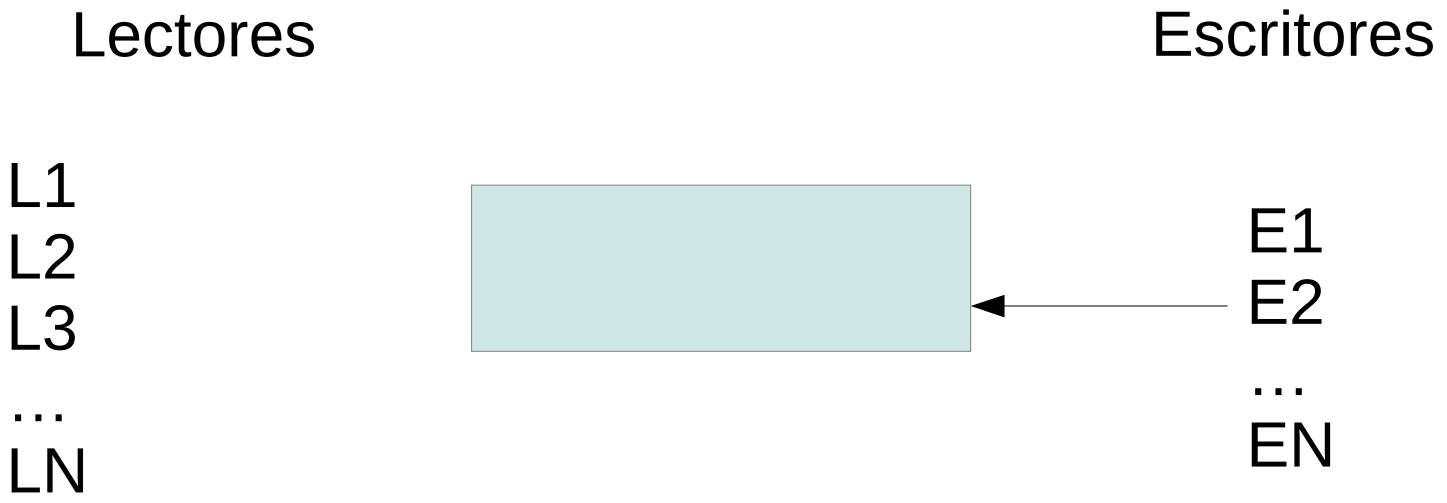
# Descripción

- Zona compartida donde hay procesos que leen y procesos que escriben.
- Los procesos que leen pueden acceder simultáneamente a la zona compartida.
- Los escritores necesitan acceso exclusivo.

# Descripción


Los lectores pueden acceder simultáneamente. Mientras acceden lectores no pueden acceder escritores.

# Descripción




Cuando un escritor accede necesita acceso exclusivo

# Ejemplos

- Cualquier información compartida con acceso concurrente y modificaciones que pueden romper la coherencia de la información:
  - Metainformación del sistema de ficheros.
  - Gestión del saldo de una cuenta bancaria.

# Prioridad

- Hay varias soluciones a este problema, según como se quiera gestionar la concurrencia:
  - Si es deseable que el sistema se pueda leer el mayor tiempo posible se prioriza el acceso de los lectores.
  - Si es deseable tener la información actualizada de forma continua se prioriza el acceso de los escritores.
- Con un número importante de lectores y/o escritores puede haber problemas de inanición.

# Prioridad para lectores

- Una vez un lector esté leyendo se permite la entrada de más lectores sin restricciones.
- Los escritores deberán esperar a que todos los lectores terminen de leer (riesgo de starvation).
- Vamos a utilizar 2 mutex, uno para bloquear la zona de lectura/escritura, y otro para un contador de lectores simultaneos:

  pthread_mutex_t excl;

  pthread_mutex_t cont_lectores;

# Prioridad para Lectores

- Lector:

```
while(1) {
    pthread_mutex_lock(&cont_lectores);
    lectores++;
    if(lectores==1) pthread_mutex_lock(&excl);
    pthread_mutex_unlock(&cont_lectores);
    leer();
    pthread_mutex_lock(&cont_lectores);
    lectores--;
    if(lectores==0) pthread_mutex_unlock(&excl);
    pthread_mutex_unlock(&cont_lectores);
}
```

# Prioridad para Lectores

- Escritor:

```
while(1) {
    pthread_mutex_lock(&excl);
    escribir();
    pthread_mutex_unlock(&excl);
}
```

# Prioridad para escritores

- Si hay un escritor esperando para escribir no se permite la entrada de más lectores.
- Para priorizar a los escritores vamos a utilizar un mutex que el primer escritor que entre a escribir va a bloquear.

  pthread_mutex_t wp;

  pthread_mutex_t excl;

  pthread_mutex_t cont_lectores;

  pthread_mutex_t cont_escritores;

# Prioridad Escritores

- Lectores: Añadimos el mutex wp sobre la solución anterior.

```
while(1) {
    pthread_mutex_lock(&wp);
    pthread_mutex_lock(&cont_lectores);
    lectores++;
    if (lectores==1) pthread_mutex_lock(&excl);
    pthread_mutex_unlock(&cont_lectores);
    pthread_mutex_unlock(&wp);
```

# Prioridad Escritores

```
    leer();
    pthead_mutex_lock(&cont_lectores);
    lectores--;
    if(lectores==0) pthread_mutex_unlock(&excl);
    pthread_mutex_unlock(&cont_lectores);
}
```

# Prioridad Escritores

- Escritores: Añadimos el mutex wp, que bloquea el primer escritor.

```
while(1) {
    pthread_mutex_lock(&cont_escritores);
    escritores++;
    if(escritores==1) pthread_mutex_lock(&wp);
    pthread_mutex_unlock(&cont_escritores);
    pthread_mutex_lock(&excl);
    escribir();
    pthread_mutex_unlock(&excl);
```

# Prioridad Escritores

```
    pthread_mutex_lock(&cont_escritores);
    escritores--;
    if(escritores==0) pthread_mutex_unlock(&wp);
    pthread_mutex_unlock(&cont_escritores);
}
```

# Prioridad Escritores: Mutex WP

- Escritor

```
while(1) {
pthread_mutex_lock(&cont_escritores);
escritores++;
if(escritores==1)
    pthread_mutex_lock(&wp);
pthread_mutex_unlock(&cont_escritores);
```

- Lector

```
while(1) {
    pthread_mutex_lock(&wp);
    pthread_mutex_lock(&cont_lectores);
    lectores++;
    if (lectores==1)
    pthread_mutex_lock(&excl);
    pthread_mutex_unlock(&cont_lectores);
    pthread_mutex_unlock(&wp);
```

Cuando un escritor bloquea wp, los lectores tienen que esperar
El resto de los escritores pueden pasar a escribir.

# Prioridad Escritores: Escritores

```
while(1) {
    pthread_mutex_lock(&cont_escritores);
    escritores++;
    if(escritores==1) pthread_mutex_lock(&wp);
    pthread_mutex_unlock(&cont_escritores);
    pthread_mutex_lock(&excl);
    escribir();
    pthread_mutex_unlock(&excl);
```

Cuando un escritor tiene wp, pueden pasar todos del bloque inicial, pero van bloqueando excl para acceder de uno en uno a la sección crítica.

# Prioridad Escritores: Desbloqueo

¿Que pasa si hay lectores y escritores intentando acceder cuando se libera wp? Vamos a partir justo después de que un escritor libere wp.

```
while(1) {

pthread_mutex_lock(&cont_escritores);

escritores++;

if(escritores==1)

    pthread_mutex_lock(&wp);

pthread_mutex_unlock(&cont_escritores);
```

Un escritor llega y avanza hasta el bloqueo de wp

```
while(1) {

    pthread_mutex_lock(&wp);

    pthread_mutex_lock(&cont_lectores);

    lectores++;

    if (lectores==1)
    pthread_mutex_lock(&excl);

    pthread_mutex_unlock(&cont_lectores);

    pthread_mutex_unlock(&wp);
```

Hay varios lectores esperando en wp

# Prioridad Escritores: Desbloqueo

```
while(1) {
pthread_mutex_lock(&cont_escritores);
escritores++;
if(escritores==1)
    pthread_mutex_lock(&wp);
pthread_mutex_unlock(&cont_escritores);
```

```
while(1) {
    pthread_mutex_lock(&wp);
    pthread_mutex_lock(&cont_lectores);
    lectores++;
    if (lectores==1)
    pthread_mutex_lock(&excl);
    pthread_mutex_unlock(&cont_lectores);
    pthread_mutex_unlock(&wp);
```

Tanto el escritor como los lectores pueden conseguir el mutex => El escritor no tiene prioridad sobre los lectores.

# Prioridad Escritores: Desbloqueo

- El escritor no tiene prioridad para obtener el mutex, por lo que es un competidor más.
- Esto es un problema justo después de que un escritor desbloquee, porque va a haber muchos lectores esperando en wp. Hasta que no se reduzca el número de lectores esperando en wp es posible que los escritores tengan problemas para bloquearlo.

# Prioridad Escritores: Desbloqueo

- En algunas librerías de mutex se garantiza que si hay algún thread/proceso esperando cuando se hace unlock lo bloquea.
- De las librerías que usamos en prácticas pthread no se comporta así, pero los semáforos POSIX sí.
- Con este comportamiento se puede añadir otro mutex para mejorar la prioridad de los escritores.

# Prioridad Escritores: Desbloqueo

- Para mitigar este problema se puede usar otro mutex en los lectores para limitar el número de lectores en wp a 1:

```
while(1) {
    lock(&lock);
    lock(&wp);
    lock(&cont_lectores);
    lectores++;
    if (lectores==1) lock(&excl);
    unlock(&cont_lectores);
    unlock(&wp);
    unlock(&lock);
```

Si hay escritores accediendo, habrá un lector esperando en wp, pero el resto estará en lock.

# Prioridad Escritores: Desbloqueo

En la misma situación, justo después de un desbloqueo de wp

```
while(1) {
lock(&cont_escritores);
escritores++;
if(escritores==1)
    lock(&wp);
unlock(&cont_escritores);
```

```
while(1) {
    lock(&lock);
    lock(&wp);
    lock(&cont_lectores);
    lectores++;
    if (lectores==1) lock(&excl);
    unlock(&cont_lectores);
    unlock(&wp);
    unlock(&lock);
```

El escritor llega a lock(wp)

Hay un lector en lock(wp), y N en lock(lock)

# Prioridad Escritores: Desbloqueo

- Aunque el lector consiga el mutex, se garantiza que cuando haga unlock(wp) después de actualizar el contador lo va a coger el escritor.
- Como los otros lectores están esperando por el mutex lock, el escritor podrá escribir en cuanto el lector que esperaba originalmente en wp termine.