Lectores/Escritores

Concurrencia y Paralelismo

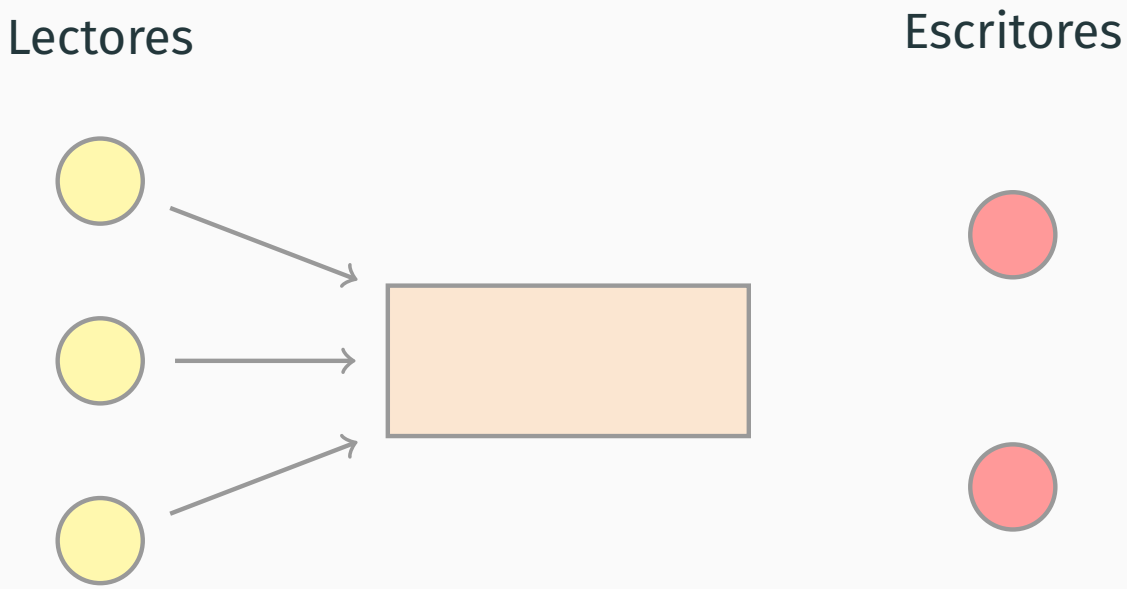
Juan Quintela
quintela@udc.es

Javier París
javier.paris@udc.es

Descripción

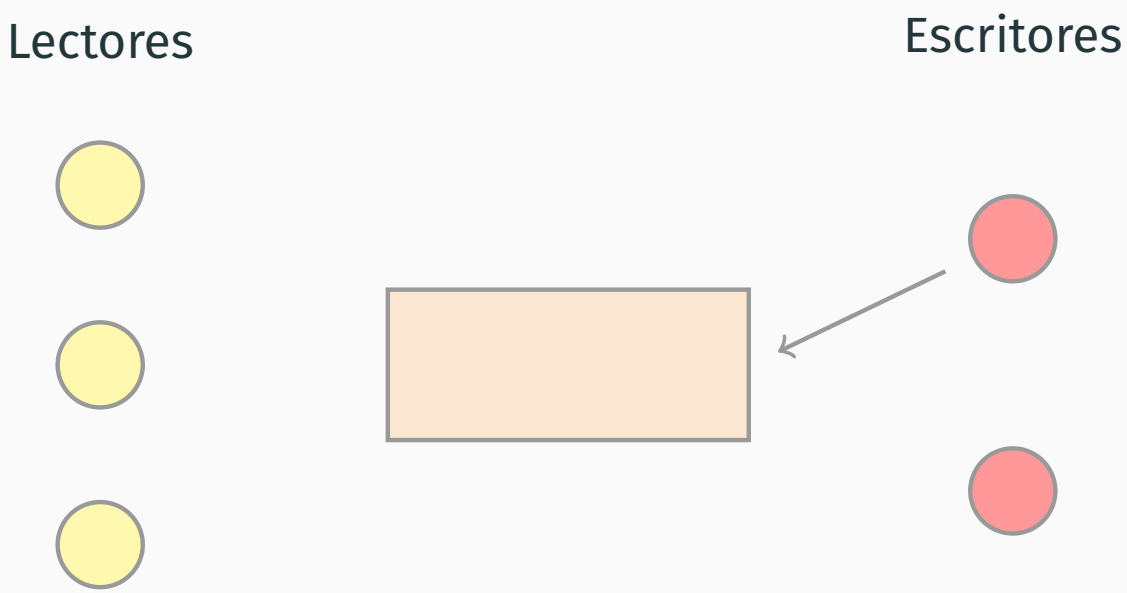
- Zona compartida donde hay procesos que leen y procesos que escriben.
- Los procesos que leen pueden acceder simultáneamente a la zona compartida.
- Los escritores necesitan acceso exclusivo.

Descripción



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

Descripción



Los escritores necesitan acceso exclusivo.

Aplicaciones

Cualquier problema con información compartida entre threads que se consulta de forma concurrente y se actualiza periódicamente:

- Metainformación del sistema de ficheros.
- Gestión de saldo de una cuenta bancaria.
- Reglas del firewall.
- Tabla de rutas.

Prioridad

- Hay dos aproximaciones en función de quien tienen prioridad para acceder a la zona compartida:
  - Si es deseable que la información esté accesible para consultas el mayor tiempo posible se priorizan los accesos de los lectores.
  - Si se prefiere que la información esté actualizada se prioriza a los escritores.
- Con un número alto de lectores y escritores se puede producir inanición.

Prioridad para Lectores

- Una vez un lector está accediendo se permite la entrada de más lectores sin restricciones.
- Los escritores deben esperar a que todos los lectores terminen (riesgo de inanición).
- Para controlar el acceso se usarán dos mutex:
  - Uno para controlar el acceso a la región compartida.
  - Otro para proteger un contador de lectores que estén accediendo a la región compartida.

Mutex

```
pthread_mutex_t excl;
pthread_mutex_t cont_lectores;
```

Prioridad para Lectores: Lector

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

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, y un contador de escritores.

Mutex

```
pthread_mutex_t wp;
pthread_mutex_t excl;
pthread_mutex_t cont_lectores;
pthread_mutex_t cont_escritores;
```

Prioridad para Escritores: Lector

Lector añadiendo el mutex 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);

  leer();

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

Prioridad para Escritores: Escritores

Escritor añadiendo el mutex wp

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

Prioridad para Escritores: Mutex WP

Escritor

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

Lector

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

Cuando un escritor bloquea wp, los lectores esperan.

El resto de los escritores pueden pasar a escribir.

Prioridad Escritores: Mutex Excl

Wp

```
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 para Escritores: Mutex WP

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

Escritor

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

Un escritor llega y avanza hasta el bloqueo de wp

Lector

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

Hay varios lectores en wp

Prioridad para Escritores: Mutex WP

Escritor

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

Lector

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

Tanto el escritor como los lectores pueden conseguir el mutex

=> El escritor no tiene prioridad sobre los lectores.

Prioridad para 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 para Escritores: Desbloqueo

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

Mutex Lectores

```
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 para Escritores: Desbloqueo

En la misma situación, justo después de desbloquearse wp.

Escritor

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

El primer escritor espera en wp. El resto están en cont_escritores.

Lector

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

Hay un lector en wp y el resto en lock.

Prioridad para Escritores: Desbloqueo

- Cuando un lector libera wp solo el lector que bloquea lock puede competir con el escritor por el mutex.
- Los otros lectores están esperando por el mutex lock.