Lectores/Escritores

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.

Los lectores pueden acceder simultáneamente. Mientras acceden lectores pueden acceder escritores. Los escritores necesitan acceso exclusivo.

Aplicaciones

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

-Metainformacion del sistema de ficheros

-Gestion de saldo de una cuenta bancaria

-Reglas de 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 este 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
mtx_t excl;
mtx_t m_readers;

//Lector
while(1) {
    mtx_lock(&m_readers);
    readers++;
    if(readers==1)
        mtx_lock(&excl);
    mtx_unlock(&m_readers);
    reader();
    mtx_lock(&m_readers);
    readers--;
    if(readers==0)
        mtx_unlock(&excl);
    mtx_unlock(&m_readers);
}

//Escritor
while(1) {
mtx_lock(&excl);
writer();
```

```
mtx_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
mtx_t wp;
mtx_t excl;
mtx_t m_readers;
mtx_t m_writers;

//Lector añadiendo el mutex wp
while(1) {
    mtx_lock(&wp);
    mtx_lock(&m_readers);
    readers++;
    if(readers==1)
        mtx_lock(&excl);
    mtx_unlock(&m_readers);
    mtx_unlock(&wp);
    leer();
    mtx_lock(&m_readers);
    readers--;
    if(readers==0)
        mtx_unlock(&excl);
    mtx_unlock(&m_readers);
}

//Escritor añadiendo el mutex wp
while(1) {
    mtx_lock(&m_writers);
    writers++;
    if(writers==1)
        mtx_lock(&wp);
    mtx_unlock(&m_writers);
    mtx_lock(&excl);
    writer();
    mtx_unlock(&excl);
    mtx_lock(&m_writers);
    writers--;
    if(writers==0)
        mtx_unlock(&wp);
    mtx_unlock(&m_writers);
}
```

Prioridad para Escritores: Mutex WP

```
//Escritor
while(1) {
    lock(&m_writers);
    writers++;
```

```
    if(writers==1)
        lock(&wp);
    unlock(&m_writers);
}

//Lector
while(1) {
    lock(&wp);
    lock(&m_readers);
    readers++;
    if(readers==1)
        lock(&excl);
    unlock(&m_readers);
    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) {
    mtx_lock(&m_writers);
    writers++;
    if(writers==1)
        mtx_lock(&wp);
    mtx_unlock(&m_writers);
    mtx_lock(&excl);
    writer();
    mtx_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. Un escritor llega y avanza hasta el bloqueo de wp. Hay varios lectores en 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.

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(&m_readers);
    readers++;
    if(readers==1)
```

```
        lock(&excl);
    unlock(&m_readers);
    unlock(&wp);
    unlock(&lock);
}
```

Si hay escritores accediendo, habrá un lector esperando en wp, pero el resto estará en lock. En la misma situación, justo después de desbloquearse wp. El primer escritor espera en wp. El resto están en m_writers. Hay un lector en wp y el resto en lock.

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.