



# Problema de lectores y escritores

Sistemas operativos

José Eduardo Rodríguez Barrios

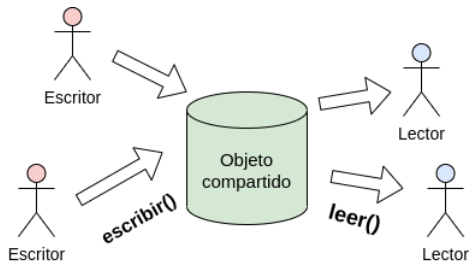
Facultad de Ingeniería, UNAM.

21 de noviembre de 2021



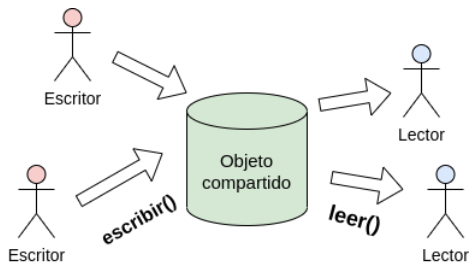
# Problema de los lectores y escritores (Courtois y colaboradores, 1971)

El problema tiene los siguientes elementos:



# Problema de los lectores y escritores (Courtois y colaboradores, 1971)

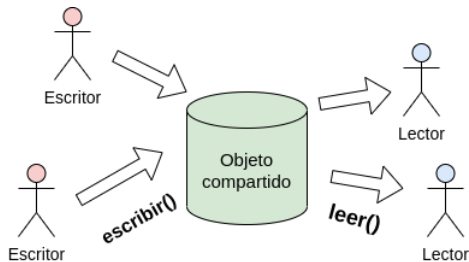
El problema tiene los siguientes elementos:



- Un objeto de datos que se va a compartir entre varios procesos concurrentes.

# Problema de los lectores y escritores (Courtois y colaboradores, 1971)

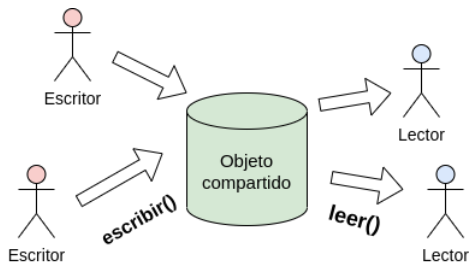
El problema tiene los siguientes elementos:



- Un objeto de datos que se va a compartir entre varios procesos concurrentes.
- Existen dos tipos de procesos:

# Problema de los lectores y escritores (Courtois y colaboradores, 1971)

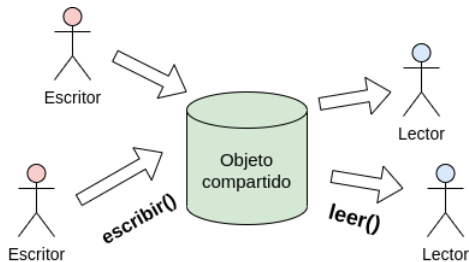
El problema tiene los siguientes elementos:



- ▶ Un objeto de datos que se va a compartir entre varios procesos concurrentes.
- ▶ Existen dos tipos de procesos:
  - ▶ **Escritores:** Los interesados en sólo querer modificar el objeto compartido.

# Problema de los lectores y escritores (Courtois y colaboradores, 1971)

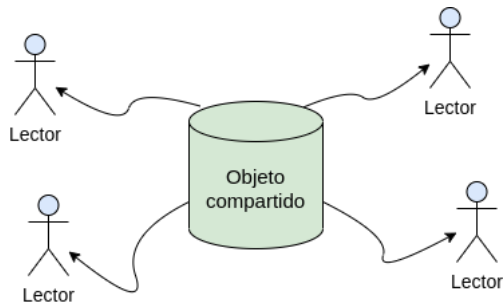
El problema tiene los siguientes elementos:



- ▶ Un objeto de datos que se va a compartir entre varios procesos concurrentes.
- ▶ Existen dos tipos de procesos:
  - ▶ **Escritores:** Los interesados en sólo querer modificar el objeto compartido.
  - ▶ **Lectores:** Los que sólo están interesados en leer el objeto compartido.

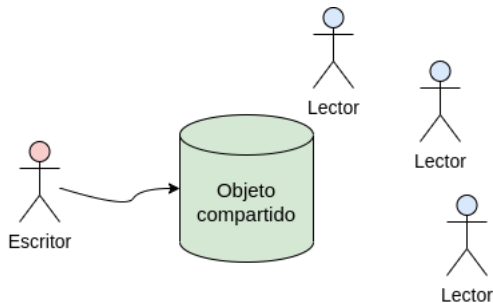
## Condiciones del problema

Cualquier número de lectores puede leer el área de datos simultáneamente.



## Condiciones del problema

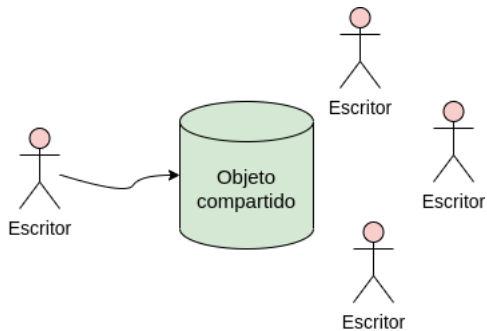
Si un escritor está accediendo al objeto compartido ningún lector puede leerlo.





## Condiciones del problema

Sólo puede escribir en el área compartida un escritor en cada instante.



## Variantes del problema

- ▶ **Primer problema de lectores y escritores:**

Exige que no se mantenga esperando a ningún lector a menos que un escritor, ya haya obtenido permiso para usar el objeto compartido.

## Variantes del problema

- ▶ **Primer problema de lectores y escritores:**

Exige que no se mantenga esperando a ningún lector a menos que un escritor, ya haya obtenido permiso para usar el objeto compartido. (prioridad a los lectores)

## Variantes del problema

- ▶ **Primer problema de lectores y escritores:**

Exige que no se mantenga esperando a ningún lector a menos que un escritor, ya haya obtenido permiso para usar el objeto compartido. (prioridad a los lectores)

- ▶ **Segundo problema de lectores y escritores**

Se requiere que, una vez que un escritor está listo, realice su escritura lo antes posible. Dicho de otro modo, si un escritor está esperando acceder al objeto, ningún lector nuevo puede comenzar leer.

## Variantes del problema

- ▶ **Primer problema de lectores y escritores:**

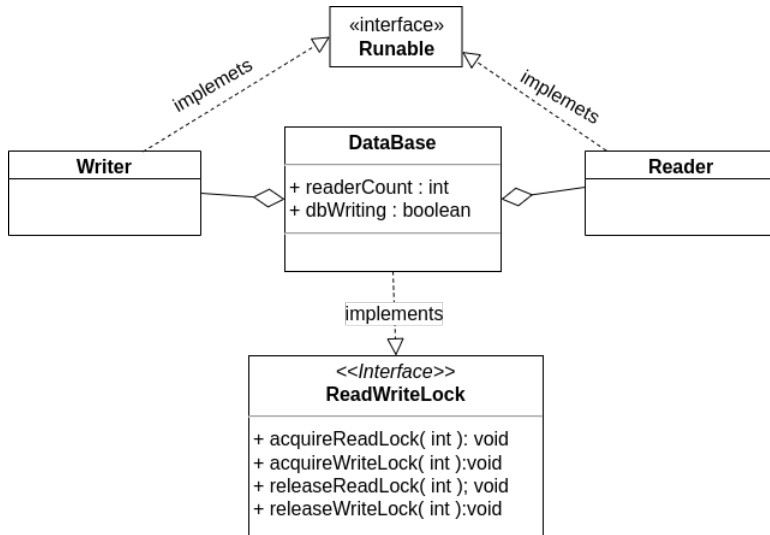
Exige que no se mantenga esperando a ningún lector a menos que un escritor, ya haya obtenido permiso para usar el objeto compartido. (prioridad a los lectores)

- ▶ **Segundo problema de lectores y escritores**

Se requiere que, una vez que un escritor está listo, realice su escritura lo antes posible. Dicho de otro modo, si un escritor está esperando acceder al objeto, ningún lector nuevo puede comenzar leer. (prioridad a los escritores)

# Solución del primer problema de lectores y escritores en Java

El programa tiene la siguiente estructura:



# Solución del primer problema de lectores y escritores en Java

## Lector

```
public class Reader implements Runnable{

    private ReadWriteLock db;
    int readerNum;

    public Reader(int readerNum, ReadWriteLock db) {
        this.db = db;
        this.readerNum = readerNum;
    }

    public void run() {
        while (true) {
            // siesta por un rato
            SleepUtilities.nap();

            System.out.println("lector " + readerNum
                               + " quiere leer.");
            db.acquireReadLock(readerNum);
            // ahora lee de la base de datos
            SleepUtilities.nap();

            db.releaseReadLock(readerNum);
        }
    }
}
```

# Solución del primer problema de lectores y escritores en Java

## Lector

```
public class Reader implements Runnable{

    private ReadWriteLock db;
    int readerNum;

    public Reader(int readerNum, ReadWriteLock db) {
        this.db = db;
        this.readerNum = readerNum;
    }

    public void run() {
        while (true) {
            // siesta por un rato
            SleepUtilities.nap();

            System.out.println("lector " + readerNum
                               + " quiere leer.");
            db.acquireReadLock(readerNum);
            // ahora lee de la base de datos
            SleepUtilities.nap();

            db.releaseReadLock(readerNum);
        }
    }
}
```

## Escritor

```
public class Writer implements Runnable{

    private ReadWriteLock db;
    int writerNum;

    public Writer(int writercount, ReadWriteLock db) {
        this.db = db;
        this.writerNum = writercount;
    }

    public void run() {
        while (true) {
            // siesta por un rato
            SleepUtilities.nap();

            System.out.println("escritor " + writerNum
                               + " quiere escribir.");
            db.acquireWriteLock(writerNum);

            // ahora escribe en la base de datos
            SleepUtilities.nap();

            db.releaseWriteLock(writerNum);
        }
    }
}
```



## Flujo de instrucciones de un lector o escritor

En ambos casos, un lector o escritor hará lo siguiente:

## Flujo de instrucciones de un lector o escritor

En ambos casos, un lector o escritor hará lo siguiente:

1. Se tomará una siesta entre 0 y 5 segundos.

## Flujo de instrucciones de un lector o escritor

En ambos casos, un lector o escritor hará lo siguiente:

1. Se tomará una siesta entre 0 y 5 segundos.
2. Luego, pedirá acceso a la base de datos ya sea para escribir o leer.

## Flujo de instrucciones de un lector o escritor

En ambos casos, un lector o escritor hará lo siguiente:

1. Se tomará una siesta entre 0 y 5 segundos.
2. Luego, pedirá acceso a la base de datos ya sea para escribir o leer.
3. Si tiene acceso, realizará su tarea entre 0 y 5 segundos. Si no esperará hasta tener acceso.

## Flujo de instrucciones de un lector o escritor

En ambos casos, un lector o escritor hará lo siguiente:

1. Se tomará una siesta entre 0 y 5 segundos.
2. Luego, pedirá acceso a la base de datos ya sea para escribir o leer.
3. Si tiene acceso, realizará su tarea entre 0 y 5 segundos. Si no esperará hasta tener acceso.
4. Una vez realizada su tarea libera la base de datos, ya sea para un escritor o lector según el caso.

## Flujo de instrucciones de un lector o escritor

En ambos casos, un lector o escritor hará lo siguiente:

1. Se tomará una siesta entre 0 y 5 segundos.
2. Luego, pedirá acceso a la base de datos ya sea para escribir o leer.
3. Si tiene acceso, realizará su tarea entre 0 y 5 segundos. Si no esperará hasta tener acceso.
4. Una vez realizada su tarea libera la base de datos, ya sea para un escritor o lector según el caso.

Esto lo hará todo el tiempo que dure en ejecución el programa.

## Lector - mecanismo de acceso

```
public synchronized void acquireReadLock(int readerNum) {  
    /**  
     * Mientras un escritor este utilizando la base  
     * de datos el lector tiene que esperar  
     */  
    while (dbWriting == true) {  
        try {  
            wait();  
        }  
        catch (InterruptedException e) { }  
    }  
    ++readerCount;  
  
    System.out.println("lector " + readerNum  
        + " está leyendo. Lectores Activos = " + readerCount);  
}
```

## Lector - mecanismo de liberación

```
public synchronized void releaseReadLock(int readerNum) {  
    --readerCount;  
    /**  
     * El último lector indica que  
     * la base de datos ya no se esta leyendo.  
     */  
    if (readerCount == 0)  
        notify();  
    System.out.println("Lector " + readerNum  
        + " ha terminado de leer. Lectores Activos = "  
        + readerCount);  
}
```



## Escritor - mecanismo de acceso

```
public synchronized void acquireWriteLock(int writerNum) {  
    /**  
     * Mientras haya lectores activos o bien un escritor  
     * esté escribiendo, éste lector se bloquea.  
     */  
    while (readerCount > 0 || dbWriting == true) {  
        try {  
            wait();  
        }  
        catch (InterruptedException e) { }  
    }  
    /**  
     * Una vez que no haya lectores ni escritores,  
     * este lector indica que está escribiendo en la base de datos.  
     */  
    System.out.println("escritor " + writerNum  
        + " está escribiendo.");  
    dbWriting = true;  
}
```

## Escritor - mecanismo de liberación

```
public synchronized void releaseWriteLock(int writerNum) {  
    dbWriting = false;  
    System.out.println("escritor " + writerNum  
        + " ha terminado de escribir.");  
    notifyAll();  
}
```

## Referencias

1. Abraham Silberschatz, Peter B. Galvin, Greg Gagne, *Operating System Concepts with Java*, Wiley, 2009.(Páginas 278-279)
2. Tanenbaum, Andrew S., *Sistemas operativos modernos*, Pearson, 2011.