UNIVERSIDAD DE GUADALAJARA

Centro Universitario De Ciencias Exactas E Ingenierías

División de Tecnologías para la Integración Ciber-Humana

Seminario de Solución de Problemas de Sistemas Operativos I7030

Reporte de Actividad Práctica

Actividad 8. Lector-Escritor

Alumnos:

Hernández Ledezma Brandon 215515031

Martínez Castillo Lisseth Abigail 218292645

Ramírez Plascencia Miguel Alejandro 215628413

Profesor: Quintanilla Moreno Francisco Javier

11 de septiembre de 2022

Objetivo de la práctica

El alumno identificará el problema del lector-escritor, así como sus diferentes aplicaciones

Antecedentes

En informática, los problemas de lectores-escritores son ejemplos de un problema común en concurrencia. Es un dilema de programación que se crea cuando varios lectores y escritores necesitan acceder al mismo recurso. Si se les permitiera el acceso a todos a la vez, podrían surgir problemas como sobreescrituras, información incompleta y otros problemas.

Algunos subprocesos pueden leer y otros pueden escribir, con la restricción de que ningún subproceso puede acceder al recurso compartido para leer o escribir mientras otro subproceso está en el acto de escribir en él. En particular, queremos evitar que más de un hilo modifique el recurso compartido simultáneamente y permitir que dos o más lectores accedan al recurso compartido al mismo tiempo. Hay varias formas de abordar el problema de lectores-escritores. Una de las soluciones más comunes implica el uso de semáforos para marcar el estado y controlar el acceso.

Problema de los primeros lectores-escritores

Supongamos que tenemos un área de memoria compartida (sección crítica) con las restricciones básicas detalladas anteriormente. En este primer problema se toma la postura de que ningún lector deberá esperar si el recurso compartido está abierto para la lectura. Esto también se llama preferencia de los lectores. En otras palabras las lecturas simultaneas son totalmente validad debido a que no modifican el recurso compartido.

Segundo problema de lectores-escritores

En este segundo problema se toma una postura muy diferente a la anterior, pues para este segundo problema se agrega la restricción de que ningún escritor, una vez agregado a la cola, se mantendrá esperando más tiempo del absolutamente necesario. Esto también se llama preferencia de los escritores. En otras palabras en el momento en el que un escritor haga una petición para escribir se le dará la oportunidad de modificar el recurso compartido independientemente si había o no lectores leyendo ese mismo recurso.

Problema de terceros lectores-escritores

De hecho, las soluciones implícitas en ambos enunciados del problema pueden resultar en inanición: la primera puede matar de hambre a los escritores en la cola, y la segunda puede matar de hambre a los lectores. Por lo tanto, a veces se propone el tercer problema de lectores-escritores, que agrega la restricción de que no se permitirá que ningún hilo se muera de hambre; es decir, la operación de obtener un bloqueo en los datos compartidos siempre terminará en un período de tiempo limitado. En otras palabras tanto las operaciones de lectura como la escritura tienen un tiempo determinado para estar en contacto con la sección critica, solucionando así bloqueos entre los lectores y escritores.

Problema de escritor de lector más simple

El problema de lector-escritor más simple que usa solo dos semáforos y no necesita una matriz de lectores para leer los datos en el búfer.

Se debe tener en cuenta que esta solución se vuelve más simple que el caso general porque se hace equivalente al problema del búfer limitado y, por lo tanto, solo N lectores pueden ingresar en paralelo, siendo N el tamaño del búfer.

Lector

do { esperar (leer) ..... leer datos ..... señal (escribir) } mientras (VERDADERO);

Escritor

do { esperar (escribir) ..... escribir datos ..... señal (leer) } mientras (VERDADERO);

Algoritmo

1. Reader se ejecutará después de Writer debido al semáforo de lectura.
2. Writer dejará de escribir cuando el semáforo de escritura llegue a 0.
3. El lector dejará de leer cuando el semáforo de lectura llegue a 0.

Desarrollo

Para el desarrollo de esta práctica se implementó una simulación donde se muestran dos listas, una lista con los lectores y una segunda lista con los escritores los cuales, se van turnando para compartir el cpu dependiendo el caso, para lectura o para escritura. Todo controlado por un semáforo que indica si es o no posible escribir. El enfoque que le hemos dado a esta simulación ha sido la postura en la que el escritor tiene preferencia sobre los lectores.

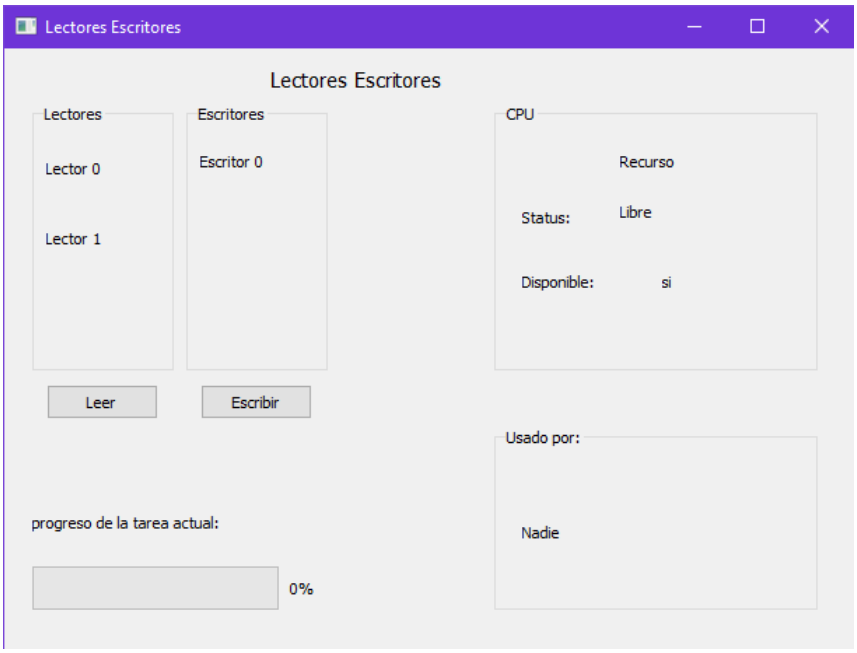
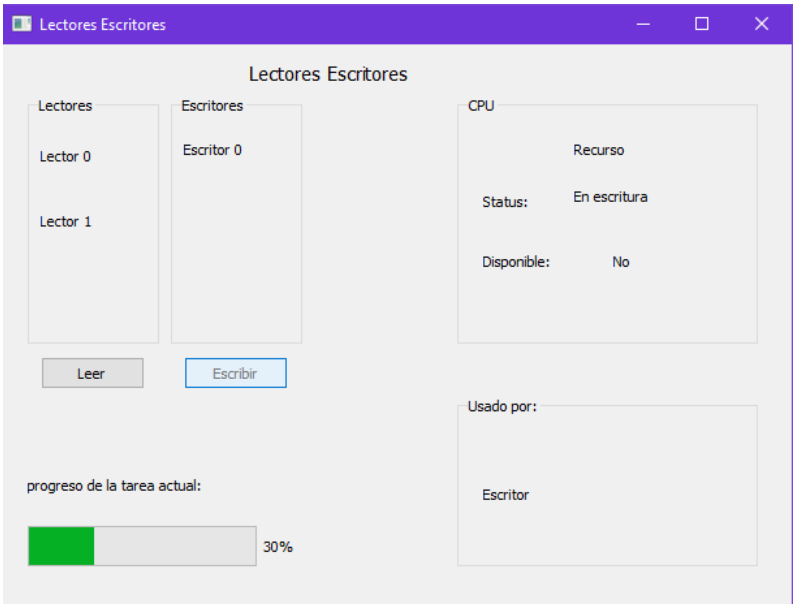


Imagen 2.1 y 2.2 Imágenes de la ejecución del programa, conforme se hacen las operaciones de lectura o escritura.



Código

```
def read():

    if widget.semaforo==0:
        index=0
        counter=0
        while (index<2):
            startThread(widget, index, 0)
            index=index+1
        for i in used_by:
            i.setText(readers[counter].text())
            counter=counter+1
        widget.status.setText("En lectura")
        widget.available.setText("No")
        widget.read.setEnabled(False)
        widget.write.setEnabled(True)
    else:
        widget.alert.setText("el recurso esta siendo utilizado\npor el Escritor por favor espere...")


def write():
    stopAll(widget)
    widget.semaforo=1
    widget.status.setText("En escritura")
    widget.available.setText("No")
    widget.read.setEnabled(True)
    widget.write.setEnabled(False)
    widget.used_by_0.setText("Escritor")
    widget.used_by_1.setText("")
    startThread(widget, 1, 1)
```

Imagen 2.3 Implementación del código para las funciones de lectura/escritura y el manejo del semáforo

```
def stopThread(widget, index):
    widget.thread[index].stop()

def stopAll(widget):
    index=0
    if(len(widget.thread)>1):
        while(index<2):
            widget.thread[index].stop()
            index=index+1
    else:
        pass



def startThread(widget, index, id):
    random=randint(10, 500)
    widget.thread[index]=threadClass(parent=None, index=index, id=id)#asignamos a una posicion del arreglo un objeto de la clase threadClass que seran nuestros hilos y
    el objeto adquiere en su atributo index su posicion dentro del arreglo a manera de identificacion
    QtTest.QTest.qWait(random)#esperamos un determinado tiempo
    widget.thread[index].start()#señalizamos que nuestro hilo iniciara su proceso
    widget.thread[index].any_signal.connect(widget.task)#
    widget.thread[index].semaforo.connect(widget.end)#
    widget.thread[index].enable.connect(widget.enable)
```

Imagen 2.4 En estas funciones se implementa la funcionalidad de los hilos, incluso, en este caso la detención de todos los hilos de lectura.

Conclusión:

Brandon Hernández Ledezma:

Esta actividad a comparación de la anterior fue un poco mas sencilla de implementar debido a la experiencia pasada, además que en este caso se eligió un enfoque un tanto sencillo, el enfoque de que los escritores toman una mayor relevancia, pues en caso de que ningún otro escritor este ya haciendo uso del recurso podía correr a los lectores que en ese momento estaban realizando lecturas simultaneas y poner en marcha su operación de escritura. Y como ya teníamos los hilos de qt bastantes controlado solo el acceso al recurso mediante un semáforo fue suficiente.

Lisseth Abigail Martínez Castillo:

Entender el funcionamiento previo del problema de productor-consumidor y de los semáforos es importante pues de esta forma tenemos contexto de cómo funciona el problema de lectores-escritores. Este problema aunque parece muy complejo de realizar, fue por otro parte un poco más sencillo de el de productor consumidor. Tal como mencionaba el uso y comprensión de los semáforos fue crucial para llevar a cabo este algoritmo de una manera sencilla, pues el semáforo fue el casi único responsable de que todo funcionara bien.

Miguel Alejandro Ramírez Plascencia:

La creación de este algoritmo en general estuvo comprendida alrededor del enfoque de que los escritores fueran los que tienen mayor prioridad, así que en este caso la implementación estaba fuertemente arraigada al uso de semáforos para que funcionara de forma excepcional. Lo cual ya más o menos habíamos contemplado mediante la elaboración y desarrollo de la anterior actividad. Debido a esto el desarrollo de esta actividad fue sin duda más o menos sencillo.

Fuentes:

- Problema de lectores y editores - frwiki.wiki. (s. f.). Recuperado 13 de octubre de 2022, de https://es.frwiki.wiki/wiki/Probl%C3%A8me_des_lecteurs_et_des_r%C3%A9dacteurs
- tok.wiki. (s. f.). Problema de lectores-escritores Problema de los primeros lectores-escritoresySegundo problema de lectores-escritores. Recuperado 13 de octubre de 2022, de https://hmong.es/wiki/Readers-writers_problem
- ¿Cuál es el problema de lectores-escritores? (s. f.). Spiegato. Recuperado 13 de octubre de 2022, de https://spiegato.com/es/cual-es-el-problema-de-lectores-escritores
- PROBLEMA DE LOS LECTORES-ESCRITORES. (s. f.). Recuperado 13 de octubre de 2022, de https://www2.infor.uva.es/%7Ecllamas/concurr/pract98/sisos30/index.html