1. Trabajo 4

**Integrantes:**

* Cristian Daniel Barbaro: [cristiandanielbarbaro@gmail.com](mailto:cristiandanielbarbaro@gmail.com)
* Leandro Ariel Pezzente: [madpirate78@yahoo.com.ar](mailto:madpirate78@yahoo.com.ar)
* Bruno Cascio: [brunocascio@gmail.com](mailto:brunocascio@gmail.com)

**Descripción del enunciado a resolver:**

Un problema muy conocido en la Programación Concurrente es el de “Lectores Escritores”, este problema consiste en un conjunto de procesos lectores que quieren leer datos de una BD y un conjunto de escritores que quieren escribir en la Base de Datos (BD). Para esto varios lectores pueden leer al mismo tiempo pero sólo un escritor puede escribir en la BD (tampoco puede haber lectores mientras el escritor está en la BD). Para simular este problema pondremos una zona común para todos los procesos que va desde la esquina (15,2) a la esquina (20,3). En esta zona cada uno de los seis procesos robotLector querrá tomar una flor y cada uno de los 3 procesos robotEscritor querrá depositar una flor de la siguiente manera:

**Robot Lector1**: se posiciona en la esquina (50,50) y debe tomar la flor de la esquina (15,2). Para tomar la flor debe consultarle al proceso robotOrganizador si no se encuentra algún robotEscritor trabajando en el area. En el caso de que el área no esté libre debe seguir pidiendo permiso para tomar la flor.

**Robot Lector2**: se posiciona en la esquina (51,51) y debe tomar la flor de la esquina (16,2). Para tomar la flor debe consultarle al proceso robotOrganizador si no se encuentra algún robotEscritor trabajando en el area. En el caso de que el área no esté libre debe seguir pidiendo permiso para tomar la flor.

**Robot Lector3**: se posiciona en la esquina (52,52) y debe tomar la flor de la esquina (17,2). Para tomar la flor debe consultarle al proceso robotOrganizador si no se encuentra algún robotEscritor trabajando en el area. En el caso de que el área no esté libre debe seguir pidiendo permiso para tomar la flor.

**Robot Lector4**: se posiciona en la esquina (53,53) y debe tomar la flor de la esquina (18,2). Para tomar la flor debe consultarle al proceso robotOrganizador si no se encuentra algún robotEscritor trabajando en el area. En el caso de que el área no esté libre debe seguir pidiendo permiso para tomar la flor.

**Robot Lector5**: se posiciona en la esquina (54,54) y debe tomar la flor de la esquina (19,2). Para tomar la flor debe consultarle al proceso robotOrganizador si no se encuentra algún robotEscritor trabajando en el area. En el caso de que el área no esté libre debe seguir pidiendo permiso para tomar la flor.

**Robot Lector6**: se posiciona en la esquina (55,55) y debe tomar la flor de la esquina (20,2). Para tomar la flor debe consultarle al proceso robotOrganizador si no se encuentra algún robotEscritor trabajando en el área. En el caso de que el área no esté libre debe seguir pidiendo permiso para tomar la flor.

**Robot Escritor1**: se posiciona en la esquina (60,60) y debe depositar una flor desde la esquina (15,2) hasta (20,2) para poder hacer esto ningún otro proceso puede estar en ninguna esquina del área compartida.

**Robot Escritor2**: se posiciona en la esquina (60,61) y debe depositar una flor desde la esquina (15,2) hasta (20,2) para poder hacer esto ningún otro proceso puede estar en ninguna esquina del área compartida.

**Robot Escritor3**: se posiciona en la esquina (60,62) y debe depositar una flor desde la esquina (15,2) hasta (20,2) para poder hacer esto ningún otro proceso puede estar en ninguna esquina del área compartida.

**Robot Organizador**: debe coordinar el acceso al área compartida por los lectores y escritores.

Para terminar el programa cada robot debe haber realizado su trabajo 3 veces. Puede asegurarse que todos los procesos tengan flores para depositar y que haya flores en las esquinas.

**Explicación de la solución presentada y comunicación entre robots:**

Para resolver el problema se utilizaron 6 robots lectores de diferentes tipos pero con una misma lógica (por limitaciones del entorno) y de la misma manera 3 robots escritores. También un robot organizador que como enuncia el problema, coordina el acceso de los robots al área compartida.

El robot organizador cuenta con el estado actual de cada robot en forma de *variables locales al robot,* es decir, lleva “semáforos” de cada uno. Es el encargado de delegar o no permisos para entrar, así como para salir, obteniendo y/o seteando en V o F el semáforo del robot que envió el mensaje.

Previamente al ingreso o egreso de cualquier robot al área, el robot que quiera acceder, deberá enviar un mensaje de evaluación del área para saber si hay escritores/lectores en la misma, luego un mensaje de acceso para decirle al organizador que encienda el semáforo correspondiente al robot en cuestión. Por último, una vez que el robot se encuentra nuevamente en su lugar de partida, le avisa al organizador la salida para que este restaure su semáforo y otros robots puedan entrar.

Muchos lectores pueden leer al mismo tiempo, sí y sólo sí no se encuentran escritores en el área compartida, pero sólo un escritor puede escribir allí si no se encuentran lectores/escritor.

Ejemplo de un lector:

Si un lector **L** desea juntar flores en una zona del área compartida, primero enviará el mensaje “permiso” al organizador **O** para que éste se lo conceda. El **O** chequeará los semáforos en ese momento y, mientras no pueda concedérselo a **L**, este último seguirá pidiendo permiso hasta que se le sea concedido.

Con el permiso garantizado, **L** enviará un mensaje de entrada al robot **O**, con el fin de que este establezca la variable del semáforo del robot **L** en *V* (robot trabajando en el área). Cuando el robot **L** termine su trabajo, volverá a la zona de su partida y avisará mediante mensajes al robot **O** su salida, y este restablecerá el semáforo del robot **L** en *F*.

Ejemplo de un escritor:

Si un escritor **E** desea depositar flores en una zona del área compartida, primero enviará el mensaje “permiso” al organizador **O** para que el mismo se lo conceda. Este chequeará los semáforos en ese momento y mientras no pueda concedérselo seguirá intentando hasta poder cedérselo a **E**.

Con el permiso garantizado, Eenviará el mensaje para entrar al robot **O**, con el fin de que este establezca la variable del semáforo del robot **E** en *V* (robot trabajando en el área). Cuando el robot **E** termine su trabajo, volverá a la zona de su partida y avisará mediante mensajes al robot **O** su salida, y este restablecerá el semáforo del robot **E** en *F*.

El trabajo del escritor consiste en depositar una flor por esquina, de manera que bloquea la esquina, deposita, libera la esquina y así continúa su trayecto.

Terminada la ejecución, el programa arroja un mensaje con el carácter **V.**

**Consideraciones para el entorno:**

*Se nos advirtió de estar usando una versión antigüa del entorno, por lo que los siguientes ítems corresponden a la versión 2.0.*

* Mejorar la ejecución de los programas, ya que a veces se queda congelado y otras no, al ejecutar el mismo código.
* Mejorar el área de texto para una mejor experiencia para el desarrollador, ya que no se permiten por ejemplo **“Ctrl + Z”.** Si se usa el comando **“Ctrl + V”** se scrollea hacia abajo y la sintaxis pasa a estar sin formato (sin colores, estilos de fuente, etc., propios del entorno).
* Poder hacer zoom del mapa, y scrolleable el estado de los robots
* Mejorar el soporte para plataforma GNU/Linux, ya que los diálogos de alerta o información a veces no se visualizan correctamente o colapsados (este último en Arch Linux, o dependientes), así como también los estilos de fuentes.
* Compilar para GNU/Linux, para no tener que usar emuladores o máquinas virtuales para correr el entorno.
* Cierres inesperados de la aplicación en algunos casos (error desconocido).
* Al guardar no pregunta si desea sobrescribir cuando se trata del mismo archivo.
* Recordar la última ruta de guardado.
* No alerta de la salida del entorno cuando se presiona la **X (ícono de cerrar ventana).**

**Explicación de las primitivas/instrucciones que se utilizaron para resolver el problema.**

*Las siguientes lineas explicadas son aplicadas para todos los robots, ya que el código contiene repetición de funcionalidades (por limitaciones del entorno).*

**ROBOT ORGANIZADOR**

robot organizador

variables

{ Identificador del robot

1..6 : Lectores

7: chequeo de robots en el área

8: aviso para entrar

9: aviso para salir

10..12: Escritores

}

ID: numero

{ Estado de los semáforos de los robots lectores }

SL1: boolean

SL2: boolean

SL3: boolean

SL4: boolean

SL5: boolean

SL6: boolean

{ Estado de los semáforos de los robots escritores }

SE1: boolean

SE2: boolean

SE3: boolean

{ Resultados para los diferentes robots lectores }

RL1: boolean

RL2: boolean

RL3: boolean

RL4: boolean

RL5: boolean

RL6: boolean

{ Resultados para los diferentes robots escritores }

RE1: boolean

RE2: boolean

RE3: boolean

op: numero { Numero de operación a realizar }

comenzar

{ repite 27 veces

Esto se hizo para que el organizador no se termine de ejecutar

ya que cada robot se ejecuta 3 veces.

6 + 3 = 9

9 x 3 = 27

}

repetir 27

recibirMensaje(ID,\*)

{

Para esta explicación se usa X para los robots lectores ( 1..6 )

}

{ Robot LectorX }

si ( ID = X )

{ Recibe la operación a realizar

(7) chequear zona

(8) aviso de ingreso

(9) aviso de egreso

}

recibirMensaje(op,LX)

si ( op = 7 )

{ Chequea que no haya escritores en la zona}

RL1 := ( SE1 | SE2 | SE3 )

mientras (RLX = V ) { Espera a que la zona quede libre }

RL1 := ( SE1 | SE2 | SE3 )

enviarMensaje(RLX,LX) { Envía el estado de los robots }

{ si el permiso es cedido, entonces recibirá la operación de ingreso}

recibirMensaje(op,LX)

si ( op = 8 ) { Entrar }

SLX := V

{ terminada la operación luego del pedido de ingreso, entonces recibirá la operación de egreso}

recibirMensaje(op,LX)

si ( op = 9 ) { Salir }

SLX := F

{

Para esta explicación se usa X para los robots escritores ( 10..12 )

NOTA: Es prácticamente análogo a un lector, salvo por la operación (7), ya que debe verificar además

que no haya lectores en el área compartida.

}

{ robot escritorX con ID = X }

si ( ID = X )

recibirMensaje(op,EX)

si ( op = 7 ) { Pedir permiso }

{ Chequea que no haya lectores/escritor en la zona compartida }

REX := ( SL1 | SL2 | SL3 | SL4 | SL5 | SL6 | SE2 | SE3 )

mientras (REX = V )

REX := ( SL1 | SL2 | SL3 | SL4 | SL5 | SL6 | SE2 | SE3 )

enviarMensaje(REX,EX) { Envía el estado de los robots }

recibirMensaje(op,EX)

si ( op = 8 ) { Entrar }

SEX := V

recibirMensaje(op,EX)

si ( op = 9 ) { Salir }

SEX := F

**ROBOT LECTOR**

*Se asume X e ID como el número de robot lector a definir ya que es análogo para los demás robots LECTORES.*

robot lectorX

variables

permisoLX: boolean { si es F entonces puede entrar, caso contrario, no.}

comenzar

{ repite 3 veces la ejecución del robot }

repetir 3

{-}

{ PIDO PERMISO PARA ENTRAR }

{-}

enviarMensaje(ID,O) { envía identificador propio }

enviarMensaje(7,O) { envía la operación para pedir permiso }

recibirMensaje(permisoLX,O)

{-}

{ concedido el permiso, va a recolectar a la esquina }

{-}

si (permisoLX = F )

{ comienza a realizar el recorrido hasta el área compartida }

repetir 3

derecha

repetir 35

mover

repetir 3

derecha

repetir 48

mover

{-}

{ ---- RECOLECTANDO :) ---- }

{-}

enviarMensaje(8,O) { Envío mensaje para entrar }

si HayFlorEnLaEsquina

tomarFlor

{ Vuelve al lugar de partida }

repetir 3

derecha

repetir 35

mover

repetir 3

derecha

repetir 48

mover

{ Avisa al organizador para salida }

{-}

enviarMensaje(9,O)

fin

**ROBOT ESCRITOR**

*Se asume X e ID como el número de robot lector a definir ya que es análogo para los demás robots ESCRITORES.*

robot escritorX

variables

permisoEX: boolean

plantadas1:numero { total de flores depositadas}

comenzar

{ repite 3 veces la ejecución del robot }

repetir 3

plantadas1 := 0 { inicializa por cada iteración, la cantidad de plantadas en 0 }

{-}

{ PIDO PERMISO PARA ENTRAR }

{-}

enviarMensaje(10,O) { envía identificador propio }

enviarMensaje(7,O) { envía la operación para pedir permiso }

recibirMensaje(permisoEX,O)

{-}

{ concedido el permiso, va a sembrar a la esquina }

{-}

si ( permisoEX = F )

{ comienza a realizar el recorrido hasta el área compartida }

repetir 3

derecha

repetir 45

mover

repetir 3

derecha

repetir 58

mover

repetir 3

derecha

{-}

{ empieza el sembrado :) }

{-}

enviarMensaje(8,O) { Envío mensaje para entrar }

mientras (plantadas1 < 6)

bloquearEsquina(PosAv,PosCa)

depositarFlor

liberarEsquina(PosAv,PosCa)

plantadas1 := plantadas1 + 1

mover

{ vuelve a su lugar de partida }

repetir 39

mover

repetir 3

derecha

repetir 58

mover

{ Avisa al organizador para salida }

enviarMensaje(9,O)

fin

**Posible enunciado**

Se trata de simular el **juego de la oca**. Para lograrlo se tienen seis robots jugadores (pueden haber más o menos jugadores) y un robot coordinador.  
Cada robot jugador se posicionará en la ciudad de la siguiente manera:

-RobotJugador1 en la esquina (1,1);

-RobotJugador2 en la esquina (2,1);

-RobotJugador3 en la esquina (3,1);

-RobotJugador4 en la esquina (4,1);

-RobotJugador5 en la esquina (5,1);

-RobotJugador6 en la esquina (6,1);

El juego finalizará cuando un robot jugador llegue primero a la calle 99 luego de avanzar por su correspondiente avenida.

Comenzará el juego el RobotJugador1 y sucesivamente hasta el RobotJugador6, cada robot jugador solicitará al robot coordinador en su turno correspondiente un número para avanzar; el coordinador simulará que tira un dado al obtener un valor aleatorio entre uno y seis y se lo devolverá al jugador que esté en su turno. El coordinaron se encarga de organizar los turnos de cada robot jugador.

Cuando el primer robot llegue al final del juego, debe informarle al coordinador que ha finalizado. El coordinador debe informar el número del robot jugador ganador.

**Conclusiones**

Debemos concluir, debido a la experiencia obtenida durante el desarrollo del programa, que las expectativas no han sido cumplidas como esperábamos. Nos hemos encontrados con varios problemas referentes al entorno que han dificultado el proceso de resolución del trabajo propuesto.

Esto nos ha obligado a tomar algunas medidas que van en contra de los fundamentos de la programación, como por ejemplo: modularización y reutilización de código.

La mayoría del tiempo dedicado al desarrollo, fue invertido en buscar alternativas implementaciones por limitaciones del entorno e incorrecta ejecución en diferentes sistemas operativos, optando por la utilización de GNU/Linux, actualmente en el que se encuentra funcionando, ya que en Windows funciona por un tiempo indeterminado, **sin concluir** ni arrojar errores.