

Práctica 2 - Repositorio Datos Asignatura

Sistemas Distribuidos

Pedro Tamargo Allué - 758267
Juan José Tambo Tambo - 755742

Implementación del algoritmo de Ricart-Agrawala en Elixir

Para implementar el algoritmo de Ricart-Agrawala, hemos dividido el algoritmo en 3 funciones y 4 procesos.

La primera función (init) hace alusión al despliegue del sistema, la creación de los procesos auxiliares: el proceso “principal” (que pide el acceso a la sección crítica), servidor mutex, servidor de variables compartidas por los procesos (debido a que no existe memoria compartida) y el proceso encargado de escuchar las peticiones (“REQUEST”) de los otros procesos que quieren entrar en la sección crítica.

La segunda función es la que se corresponde con el pre-protocolo (begin_op), en la cual se envían las peticiones a los procesos “REQUEST” de los otros nodos, y se espera la confirmación como una tupla {ok, pid}, siendo pid el PID del proceso REQUEST que nos ha autorizado a acceder a la sección crítica. En esta función se accede a ciertas variables compartidas como el estado del proceso y el reloj lógico escalar, por lo tanto, mediante el servidor mutex nos aseguramos de acceder y modificar las variables compartidas en exclusión mutua. La implementación de estas esperas se ha hecho mediante la sentencia receive del lenguaje, y para la implementación de las operaciones sobre listas se ha utilizado la función predefinida Enum.map(lista, funcion).

La tercera función es la que se encarga del post-protocolo (end_op), en la cual, actualizamos el estado del proceso a “out” (fuera de la sección crítica), y mediante el servidor de variables obtenemos los procesos encolados en la cola de espera y les enviamos el permiso (utilizando la tupla {ok,pid}, siendo pid el PID del proceso REQUEST de nuestro nodo).

El proceso mutex es un servidor que proporciona el acceso en exclusión mutua a ciertas instrucciones como las que acceden al servidor de variables para gestionar las variables compartidas.

El servidor de variables es el encargado de mantener actualizados los valores de las variables compartidas a los procesos. Las variables son: el estado del proceso principal respecto a la sección crítica, el reloj lógico escalar del proceso y la cola de procesos en espera.

El proceso REQUEST ejecuta la función “receive_petition”, la cual se encarga de procesar las peticiones de los procesos principales que quieren acceder a la sección crítica. Se determina si un proceso tiene más prioridad que otro mediante la variable “prio”, cuyo valor se decide en función del estado (del proceso “principal” que se ejecuta en el nodo) respecto a la sección crítica, el reloj lógico escalar de nuestro nodo y del nodo que nos envía la petición y de la matriz de exclusión “exclude” (que permite determinar si dos procesos pueden estar a la vez en la SC dependiendo de su tipo de operación, p.ej: dos lectores pueden estar en la SC a la vez, pero un lector y un escritor no pueden coincidir en la SC a la vez). Una vez determinada la prioridad, si nuestro proceso principal tiene prioridad sobre el otro proceso, añadiremos a la cola el proceso que nos ha enviado la petición. En el caso de que no tengamos la prioridad, enviaremos la tupla {ok, pid} (siendo pid nuestro el PID, ya que somos el proceso REQUEST).

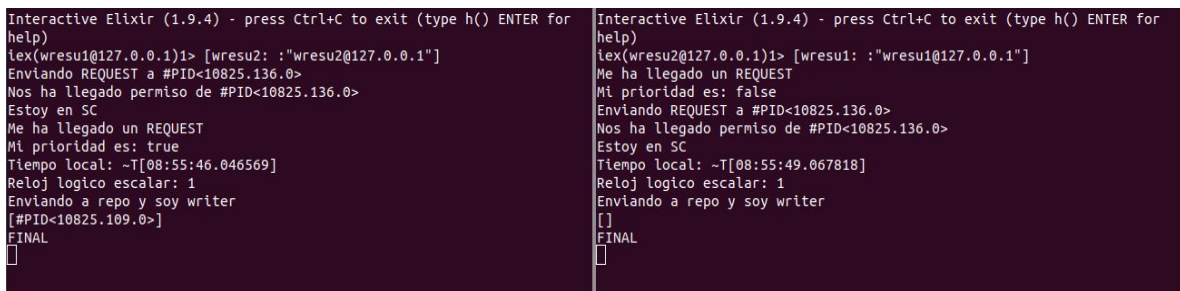
El proceso principal es el que querrá acceder a la SC, y para ello utilizará las funciones `init`, `begin_op` y `end_op`.

Pruebas sobre la corrección del algoritmo implementado

Para la realización de las pruebas, se ha utilizado el módulo Repositorio proporcionado en junto con el enunciado de la práctica. Sobre este módulo, que almacena información acerca del enunciado de un trabajo de una asignatura. Este enunciado se divide en: resumen, parte principal y entrega. De cara al trabajo con este módulo hemos establecido la granularidad del mismo (capacidad de acceso en exclusión mutua a los distintos campos del repositorio) como mínima, es decir, no puede convivir en la sección crítica más de un proceso si existe un proceso de tipo “escritor” en ella. De la misma forma, pueden existir más de un proceso “lector” en la sección crítica si no hay ningún proceso “escritor” en ella.

Para la realización de las pruebas, se ha utilizado el módulo “repositorio” que se proporcionaba en moodle. Se ha considerado de mayor relevancia las siguientes pruebas:

- **Dos escritores:** Lanzando dos escritores sobre la misma parte del repositorio (en este caso resumen), se han obtenido los siguientes resultados:



```
Interactive Elixir (1.9.4) - press Ctrl+C to exit (type h() ENTER for help)
iex(wresu1@127.0.0.1)1> [wresu2: : "wresu2@127.0.0.1"]
Enviando REQUEST a #PID<10825.136.0>
Nos ha llegado permiso de #PID<10825.136.0>
Estoy en SC
Me ha llegado un REQUEST
Mi prioridad es: true
Tiempo local: ~T[08:55:46.046569]
Reloj logico escalar: 1
Enviando a repo y soy writer
[#PID<10825.109.0>]
FINAL
█

Interactive Elixir (1.9.4) - press Ctrl+C to exit (type h() ENTER for help)
iex(wresu2@127.0.0.1)1> [wresu1: : "wresu1@127.0.0.1"]
Me ha llegado un REQUEST
Mi prioridad es: false
Enviando REQUEST a #PID<10825.136.0>
Nos ha llegado permiso de #PID<10825.136.0>
Estoy en SC
Tiempo local: ~T[08:55:49.067818]
Reloj logico escalar: 1
Enviando a repo y soy writer
[]
FINAL
█
```

Captura de pantalla resultante de la ejecución de dos procesos escritor: w1(izq.) y w2 (der) sobre el atributo resumen.

Ambos procesos intentan acceder al mismo tiempo a la SC, pero al ser ambos escritores, esto no es posible. Por ello, se observa que w1 envía una solicitud a w2 en primer lugar, por lo que w2 le envía una respuesta para que pueda acceder a la SC. Una vez dentro, w1 recibe la petición de w2, así que lo introduce a la lista de procesos en espera.

Cuando está fuera de la SC, envía permiso a todos procesos que tiene en la lista de espera, que en este caso únicamente es el w2, el cual recibe esta confirmación y entra en SC. Se muestran los tiempos dentro de la SC para saber cuándo acceden a ella, haciendo visible que no entran en el mismo instante.

- **Escritores y Lectores:** En esta prueba, se han ejecutado dos procesos lectores: uno sobre principal y otro sobre entrega, y dos procesos escritores, también sobre principal y entrega.

```

Enviando REQUEST a #PID<10827.140.0>
Enviando REQUEST a #PID<10829.145.0>
Enviando REQUEST a #PID<10825.146.0>
Me ha llegado un REQUEST
Nos ha llegado permiso de #PID<10827.140.0>
Mi prioridad es: true
Me ha llegado un REQUEST
Mi prioridad es: true
Nos ha llegado permiso de #PID<10829.145.0>
Nos ha llegado permiso de #PID<10825.146.0>
Estoy en SC
Me ha llegado un REQUEST
Mi prioridad es: false
Tiempo local: ~T[10:03:44.178782]
Reloj logico escalar: 2
Enviando a repo y soy lector
Valor de read_principal es Elixir.Modifcio_principal_1
[#PID<10827.109.0>, #PID<10829.109.0>]
[]
FINAL
Me ha llegado un REQUEST
Mi prioridad es: false
Me ha llegado un REQUEST
Mi prioridad es: false
Me ha llegado un REQUEST
Mi prioridad es: false
Enviando REQUEST a #PID<10827.140.0>
Enviando REQUEST a #PID<10829.145.0>
Enviando REQUEST a #PID<10825.146.0>
Nos ha llegado permiso de #PID<10825.146.0>
Nos ha llegado permiso de #PID<10829.145.0>
Nos ha llegado permiso de #PID<10827.140.0>
Estoy en SC
Tiempo local: ~T[10:03:53.245827]
Reloj logico escalar: 2
Enviando a repo y soy lector
Valor de read_entrega es Elixir.Modifcio_entrega_1
[]
FINAL
]
Enviando REQUEST a #PID<10829.145.0>
Me ha llegado un REQUEST
Enviando REQUEST a #PID<10825.146.0>
Mi prioridad es: false
Enviando REQUEST a #PID<10827.146.0>
Me ha llegado un REQUEST
Mi prioridad es: false
Nos ha llegado permiso de #PID<10827.146.0>
Me ha llegado un REQUEST
Mi prioridad es: true
Nos ha llegado permiso de #PID<10825.146.0>
Nos ha llegado permiso de #PID<10829.145.0>
Estoy en SC
Tiempo local: ~T[10:03:50.219513]
Reloj logico escalar: 2
Enviando a repo y soy writer
[#PID<10827.109.0>]
FINAL
Enviando REQUEST a #PID<10829.140.0>
Enviando REQUEST a #PID<10825.146.0>
Enviando REQUEST a #PID<10827.146.0>
Me ha llegado un REQUEST
Mi prioridad es: false
Me ha llegado un REQUEST
Mi prioridad es: true
Nos ha llegado permiso de #PID<10829.140.0>
Nos ha llegado permiso de #PID<10827.146.0>
Me ha llegado un REQUEST
Mi prioridad es: true
Nos ha llegado permiso de #PID<10825.146.0>
Estoy en SC
Tiempo local: ~T[10:03:47.195192]
Reloj logico escalar: 2
Enviando a repo y soy writer
[#PID<10829.109.0>, #PID<10827.109.0>]
[]
FINAL

```

Captura de pantalla resultante de ejecutar dos procesos lectores: lprin (arriba iz), lentre(abajo iz), y dos procesos escritor: wprin(arriba der), wentre(abajo der). Ya se había escrito sobre ese repositorio previamente.

En este caso, se observa que el primer procesos en entrar en la SC es lprin, el cual muestra por pantalla lo que existía en la zona “principal” del repositorio y envía confirmación a los procesos que tenía en la lista de espera. El proceso wentre, recibe esta confirmación y accede a la SC, modificando así el campo de “entrega” y una vez fuera, envía permiso a los otros dos procesos que aún no han conseguido entrar. Esta ejecución se repite en los dos últimos procesos.

Una vez analizados los resultados, se puede observar que los procesos writer acceden a la SC en exclusión mútua, ya que mientras están en ella, ni los procesos lector ni escritor pueden acceder.