I.T.C.R.

Sistemas Operativos

Departamento de Computación
Prof: Ing. Erika Marín Schumann.

Aplicación Readers – Writers
II Semestre del 2016

Objetivos del proyecto

- Implementar el problema de los Reader & Writers.
- Implementar la sincronización de procesos.
- Profundizar el conocimiento de Semáforos.
- Implementar un esquema de memoria compartida
- Conocer algunas funciones de Linux para la sincronización y memoria compartida.

Definición general

Este primer proyecto tiene como finalidad implementar el problema de los Readers & Writers. Para esto se debe implementar cada uno de estos como programas separados. El problema incluirá un nuevo tipo de Reader. Que nominaremos el "Reader Egoísta". Este será un tipo de proceso que contrario a los demás readers no permite que otros procesos estén accediendo la memoria compartida. Además lo que leen no quieren que nadie más lo vean; por lo que al leer se llevan consigo la información.

Como bien es sabido este problema incluye algún tipo de sincronización. Existen diferentes tipos de semáforos. Se debe tomar en cuenta que dado que en este problema los procesos a utilizar son programas separados, se debe buscar el tipo de semáforo que me permite la comunicación entre programas.

Este problema incluye un archivo compartido entre todos los actores. Se debe utilizar la memoria compartida junto con la una función de memoria mapeada a disco para compartir este archivo entre los programas.

Para implementar este problema se deben crear 6 programas que se detallan a continuación.

Descripción Detallada

- Programa Inicializador: Este programa se encarga de crear el ambiente. Pide los recursos y los inicializa de manera que los actores encuentren todo listo en el momento de empezar a funcionar. De parámetro debe obtener la cantidad de vectores o líneas que tendrá la memoria compartida. Este proceso debe morir después de realizar la inicialización.
- Programa Writer: Este es el programa que crea los procesos escritores, por lo que necesita de un parámetro que le especifique cuantos escritores va a haber en escena. Al crear la cantidad de escritores (hilos) que el usuario solicite, éstos, de manera secuencial en el archivo, irán escribiendo su mensaje. Los escritores son procesos dedicados, esto significa que ningún lector u otro escritor puede tener acceso al archivo mientras un escritor esté escribiendo. El escritor solo podrá escribir en espacios libres del archivo. No puede escribir si el archivo está lleno. La línea donde debe escribir es la siguiente vacía. Después de escribir el Writer se dormirá por una cantidad de segundos. Esta cantidad de segundos es un parámetro de este programa, al igual que el tiempo que estará "escribiendo" Pero durante el momento en que duermen el archivo queda libre para que otro proceso lea o escriba.
- Mensaje: El mensaje debe caber en un registro o línea del archivo. Y debe contener la siguiente información: PID del proceso que escribe. Hora y Fecha en que lo escribe. Y línea del archivo en que se está escribiendo.
- Programa Reader: Este es el programa que crea los procesos lectores, por lo que necesita de un parámetro que le especifique cuantos lectores va a haber en escena. Al crear la cantidad de lectores que el usuario solicite, estos de manera secuencial en el archivo irán leyendo los mensajes que los escritores van poniendo. Los readers leen de manera secuencial, siempre y cuando haya un mensaje en la siguiente entrada. Cada vez que cada Reader llega a la última línea del archivo, vuelve a iniciarlo. Cada lector debe saber por donde va en el archivo. Si hay un lector leyendo del archivo, los demás readers pueden leer también siempre que haya algo que leer por donde ellos van leyendo. Lo que los readers duren leyendo se definirá como un parámetro del programa, así como el tiempo en que duerman.

- Programa Reader Egoísta: Este es el programa que crea los procesos lectores egoístas, por lo que necesita de un parámetro que le especifique cuantos lectores de este tipo va a haber en escena. Al crear la cantidad de lectores egoístas que el usuario solicite, éstos irán seleccionando de manera aleatoria entradas en el archivo, si la entrada que seleccionaron tiene algún mensaje lo leen y como son egoístas no quieren que nadie más lo lea, por decirlo así se lo roban. Cuando un lector de este tipo accede al archivo nadie más puede estar leyendo o escribiendo. Los procesos readers egoístas duran cierta cantidad de tiempo leyendo, dejan que alguien más acceda al archivo y duermen por un rato. La cantidad de tiempo leyendo y durmiendo es un parámetro de entrada al programa. Este tipo de reader tiene prioridad sobre los readers normales. Pero no sobre los writers. Están al mismo nivel. No permitir que más de 3 readers egoístas tengan acceso al archivo de manera consecutiva. Si se presenta esta situación el archivo debe ser entregado a algún otro proceso que esté compitiendo por él. Los Writers tienen prioridad. SI no hay nadie más podrán seguir leyendo los egoístas hasta que alguien solicite el recurso.
- Programa Finalizador: Se encarga de matar todos los procesos que estén en escena. Devolver los recursos que había solicitado. Y eliminar el archivo físico.
- Programa Espía: Para efectos de control este programa debe responder a las siguientes solicitudes del usuario
  - Estado del archivo en determinado momento. Deben verse todas las líneas, aunque no tengan nada.
  - Estado de los Writers
  - Estado de los Readers
  - Estado de los Reader Egoístas
  - Al pedir el estado debe decirme
    - El PID de los proceso que tenga acceso al archivo en ese momento
    - El PID de los procesos que estén dormidos (writers y readers egoístas)
    - El PID de los procesos que estén bloqueados
- Sincronización de Procesos. Debe utilizar Semáforos. Asegúrese de escoger los semáforos adecuados.
- Memoria Mapeada a Disco: No utilice mmap. Utilice la instrucción schmget, que me permite compartr segmentos de memoria.
- BITACORA: Se debe registrar una bitácora de todas las acciones que se llevaron a cabo durante el proceso. Esto incluye que diga por cada PID que hizo (leer o escribir), que tipo es (reader, writer, o reader egoísta), hora y el mensaje que leyó o escribió. Esta bitácora debe quedar en un archivo. Y se puede consultar al final del proceso.

Documentación

Se espera que sea un documento donde especifique:

- El tipo de semáforos que utilizó y porqué?
- Una explicación de cómo logró la sincronización?
- ¿Qué hubiera sido diferente si hubiera utilizado mmap?
- El análisis de resultados del programa junto con unos casos de pruebas. El Análisis debe resumir el resultado de la programación, qué sirve, qué no sirve y aspectos que consideren relevantes.
- Cómo compilar y ejecutar los programas.

Aspectos Administrativos

- El desarrollo de este programa debe de realizarse en grupos de exactamente dos personas salvo acuerdo con el profesor.
- El trabajo se debe de entregar por correo el día 21 de noviembre de 2016 antes de media noche.
- Deben entregar el código fuente junto con el ejecutable. La documentación se debe de entregar IMPRESA el día de las revisiones.
- Recuerde que oficialmente no se recibirán trabajos con entrega tardía.