I.T.C.R.

Departamento de Computación
Prof: Ing. Erika Marín Schumann.

Sistemas Operativos

Simulación Particiones Dinámicas
I Semestre del 2019

Objetivos del proyecto

- Implementar un problema de sincronización.
- Profundizar el conocimiento de Semáforos.
- Conocer algunas funciones de Linux para la sincronización.

Definición general

Este segundo proyecto tiene como finalidad implementar un problema de sincronización de procesos en una simulación de asignación de memoria a procesos mediante los diferentes algoritmos de particiones dinámicas. Para esto se debe implementar cada uno de los programas que se listan abajo como programas separados. Los programas deben ser implementados en C.

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 para llevar una bitácora de eventos. Se debe utilizar la memoria compartida para simular la memoria a asignar a los procesos. Si necesitan otro tipo de memoria compartida para la información del problema pueden usarla.

Para implementar este problema se deben crear 4 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. Este programa pide la cantidad de líneas o espacios de memoria que va a haber y solicita la memoria compartida al sistema operativo. Este proceso debe morir después de realizar la inicialización.
- Programa Productor de Procesos: Este es un programa que genera procesos que llegan al “Ready” (Threads). Lo primero que debe hacer es solicitar el tipo de algoritmo con el que se desea correr la simulación (Best-Fit, First-Fit, Worst-Fit). Para cada uno de los hilos se define de manera Random el tamaño o cantidad de líneas (1-10) y el tiempo (20s-60s). La Distribución con la que se generaran los procesos estará determinada por un tiempo aleatorio (30s-60s). Cada proceso debe buscar dentro de la memoria un espacio donde entrar de forma completa. Si no hay espacio el proceso muere. Solo un proceso a la vez puede estar corriendo el algoritmo de buscada de espacio a la vez, para que no haya choques que 2 procesos seleccionen el mismo hueco. Una vez que haya encontrado lugar transcurrirá un sleep con la cantidad de tiempo definida para él y después de forma exclusiva (Región Critica) devuelve los espacios de memoria que tenía asignados.
- 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. Para las que están ocupadas debe decir el proceso que las ocupa.
  - Estado de los Procesos
  - Al pedir el estado debe decirme
    - El PID de los proceso que tenga acceso a la memoria en ese momento
    - El PID de los procesos que estén “ejecutando”
    - El PID de los procesos que estén bloqueados
- Programa Finalizador: Se encarga de matar todos los procesos que estén en escena. Devolver los recursos que había solicitado. Y cerrar el archivo de la Bitácora.
- Sincronización de Procesos. Debe utilizar Semáforos. Asegúrese de escoger los semáforos adecuados.

- Bitácora: Todos los procesos deben registrar sus acciones. Esto incluye que diga por cada PID, la acción, que tipo es (asignación, desasignacion), hora y líneas asignadas. Para el caso que un proceso no haya entrado a la memoria debe registrar este hecho en la bitácora.
- Interfaz: No es necesario que implementen una interfaz gráfica complicada (No se van a dar puntos de extra por esto, no se compliquen.) Lo que espero es que por medio de texto se vea el movimiento de los procesos. Y se pueda identificar el estado que tiene esta en todo momento.

Documentación

Se espera que sea un documento donde especifique lo siguiente:

a) Portada, índice, introducción
b) Estrategia de Solución :
    - El tipo de semáforos que utilizó y porqué?
    - Una explicación de cómo logró la sincronización?
c) Análisis de Resultados:
    - Deberá elaborar un listado de todas y cada una de las actividades y tareas que deben cubrirse a nivel funcional, para cada una de ellas debe aportar el porcentaje de realización y en caso de no ser el 100% debe justificarse.
    - Análisis de cuál algoritmo genera más fragmentación?
d) Lecciones aprendidas: Debe prepararse un listado de las lecciones aprendidas producto del desarrollo de la tarea programada. Las lecciones aprendidas pueden ser de carácter personal y/o técnico que involucre aspectos que han logrado un aprendizaje en temas de investigación, desarrollo de habilidades técnicas y habilidades blandas como trabajo en equipo, comunicación, forma de expresar ideas, entre otros.
e) Casos de pruebas: se espera que definan claramente cada prueba, cuáles son los resultados esperados y cuáles fueron los resultados obtenidos. No es necesario que sean grandes pero deben evaluar la funcionalidad completa del programa.
f) Comparación: Investigar la diferencia entre mmap y shmget?.
g) Manual de usuario: especificar como compilar y correr su tarea.
h) Bitácora de trabajo durante las semanas de trabajo, incluyendo verificaciones realizadas (si existieran) de consultas realizadas con el profesor o asistente.
i) Bibliografía y fuentes digitales utilizadas

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 el día 29 de mayo de 2019 antes de las 12 media noche, por correo o tec digital. (La documentación debe ir en el correo)
- Deben entregar el código fuente junto con el ejecutable.
- Los trabajos que se entreguen de forma tardía serán evaluados en base a 60