TEC | Tecnológico de Costa Rica

Instituto Tecnológico de Costa Rica
Área Académica de Ingeniería en Computadores
CE 4303 — Principios de Sistemas Operativos

Proyecto 1
Constructor de textos

Fecha de asignación:	20 de marzo, 2024	Fecha de entrega:	19 de Abril, 2024
Grupos:	2 personas	Profesor:	Jason Leitón Jiménez

1. Objetivo

Aplicar las técnicas de sincronización eficientes, con el fin de solucionar problemas con los recursos compartidos entre dos o más procesos.

2. Atributos a evaluar

- Aprendizaje continuo. Se requiere que el estudiante valore las estrategias y el conocimiento adquirido para alcanzar el objetivo.
- Herramientas de Ingeniería. Se requiere que el estudiante sea capaz de adaptar técnicas, recursos y herramientas modernas para la solución de problemas.

3. Motivación

El problema del consumidor y productor es un clásico de los conflictos en los ámbitos de los sistemas operativos, muchos de los problemas que se encuentran en la informática y otras áreas se puede modelar de la misma manera, y con ello se podría solucionar aplicando las mismas técnicas. La idea fundamental de este proyecto es comunicar *heavy process* a través de memoria principal sin utilizar *busy waiting*. Esta sección de memoria deberá ser inicializada por un proceso.

4. Descripción

En el proyecto existen 4 tipos de procesos (*Heavy Process*), los cuales son descritos a continuación:

4.1. Creador

El creador es el proceso encargado de inicializar el espacio de memoria compartida de acuerdo con la cantidad de caracteres que se desean compartir simultáneamente. Esta cantidad es ingresada por el usuario utilizando la consola. Es importante mencionar que el espacio debe ser justo lo necesario, por ejemplo, si el usuario coloca 100 caracteres, se deberá crear en memoria

principal 100 campos en donde cada uno tendrá el tamaño que corresponda, en caso de que sea de tipo entero será 4 bytes, para un total de 400 bytes. Se sugiere tratar la memoria como un vector y utilizar aritmética de punteros. Ya que una de las características del proyecto es que la sincronización y comunicación entre procesos sean eficientes.

Una vez que la memoria esté lista para utilizarse, el proceso creará la visualización en "tiempo real"del contenido de la memoria compartida.

4.2. Cliente

Este proceso será capaz de acceder a la memoria compartida inicializada por el proceso creador. El cliente tomará los caracteres de un archivo de texto (indicado por el usuario a través de la consola, de cualquier tamaño) y los colocará de manera circular en la memoria compartida. Cada caracter debe ser almacenado en un campo. Esto significa que si el proceso creador inicializó 100 espacios, entonces se podrá almacenar 100 caracteres de manera simultánea. Cada vez que se introduce un elemento, el cliente debe mostrar en pantalla el caracter, la hora, y la posición donde fue insertado.

La inserción de los caracteres se realizara de manera secuencial cada vez que el usuario presione "Enter.º de manera automática (posteriormente se detallan los modos). Cuando se haya transferido la totalidad de elementos en el archivos, el cliente deberá establecer en "Trueüna variable global, la cual tendrá que acceder el servidor.

Es importante destacar que no se permite el uso de *busy waiting* ni *sockets* por parte del cliente. Además se debe considerar que la inserción sea de manera **circular**.

En caso de que no haya espacio en la memoria compartida para introducir más caracteres, este proceso entra en un estado de bloqueado hasta que se cuente con espacio para volver a introducir datos. Es de suma importancia no perder ningún caracter ni el orden de estos.

El cliente debe mostrar en pantalla el contenido inicial del archivo y mostrar cuales carecteres se están introduciendo en memoria compartida. Puede ser marcándolos o eliminándolos.

Cabe destacar, que pueden existir n instancias de este proceso, es decir, se puede crear varios clientes al mismo tiempo. Se debe proporcionar un mecanismo para para terminar, de manera elegante, el proceso.

4.3. Re-constructor

Este módulo corresponde a otro *heavy process* cuya función es tomar los caracteres de memoria compartida y reconstruir el archivo original. Este proceso siempre trata de leer desde memoria principal (compartida), en caso de que no se tenga nada que leer entrará en un estado de bloqueado hasta que existan elementos por leer. Cabe destacar que no se permite el uso de *busy Waiting*.

De manera similar al cliente se leerá un caracter cada vez que el usuario presione un "Enter.º de manera automática. Cada vez que se lea un elemento se deberá imprimir la posición (Campo)

donde estaba, la fecha y hora que fue insertado y el valor del dato.

Este proceso creará una pantalla donde se muestre el proceso de reconstrucción del archivo. Se debe de observar como se agrega caracter por caracter.

El Re-Constructor terminará de forma elegante cuando reciba una señal del cliente indicando que ya no hay caracteres por enviar. Se debe proporcionar un mecanismo para que todos los reconstructores terminen al mismo tiempo.

Cabe destacar que pueden existir varias instancias de este proceso.

4.4. Estadísticas

Este módulo será el encargado de mostrar las estadísticas una vez que cierren tanto los clientes como los Re-constructores. Se recomienda contar con una sección de memoria compartida para guardar únicamente los datos relacionados con las estadísticas, los cuales son:

1. Tiempo bloqueado del cliente.
2. Tiempo bloqueado del Re-constructor.
3. Caracteres transferidos y los que quedan en el buffer.
4. Espacio total de memoria utilizado.
5. Tiempo total en modo usuario y kernel del cliente y del Re-constructor.

4.5. Modos de ejecución

Tanto el cliente como el reconstructor podrán ejecutarse en dos modos distintos. Esto se debe indicar en el momento de crear la instancia del proceso (en el comando de ejecución por medio de un parámetro). Los modos serán los siguientes:

1. Manual: Se lee o se escribe en el buffer cada vez que se presiona enter.
2. Automático: Se lee o se escribe en el buffer cada ciento tiempo, el cual debe ser especificado en el comando de ejecución cuando se crea los clientes o los reconstructores.

5. Requerimientos técnicos

- Este proyecto se debe realizar en el lenguaje de programación C.
- Debe ser implementado en Linux (no máquina virtual) y se debe proporcionar un makefile.
- No se permite soluciones “alambradas”.

- Se debe prestar especial atención a los errores de acceso a memoria o utilización de recursos. Es inaceptable el error *segmentation fault* o *core dumped*.
- No se permite busy waiting como se mencionó anteriormente.

6. Documentación- Estilo IEEE-Trans (máximo 5 páginas)

- Introducción: Teoría necesaria, breve descripción del proyecto y qué es lo que se espera en el escrito.
- Ambiente de desarrollo: Configuración básica se debe utilizar para ejecutar el proyecto. Frameworks, bibliotecas externas o principales, aplicaciones de terceros, herramientas de desarrollo.
- Atributos: Esta sección deben de describirse cuales atributos fueron reforzados durante el desarrollo del proyecto. Para el atributo de **aprendizaje continuo** debe responder las siguientes preguntas:
  - ¿Cuales son las necesidades actuales de aprendizaje para enfrentar el proyecto?
  - ¿Cuáles son las tecnologías que se pueden utilizar para el desarrollo?
  - ¿Cuáles acciones se implementó para el desarrollo del proyecto (organización de tiempo, búsqueda de información, repaso de contenidos, entre otros)?
  - Evalúe de forma crítica la eficiencia de las acciones implementadas en el contexto tecnológico.
- Detalles del diseño del programa desarrollo, tanto del software como del hardware (en caso de que aplique): Diagramas de flujo, imágenes, descripciones entre otros, todo lo que sea necesario para entender de una mejor manera el diseño y funcionamiento del proyecto. Es necesario que realice como mínimo, el diagrama de arquitectura, componentes y secuencia.
- Instrucciones de cómo se utiliza el proyecto.
- Tabla de actividades por cada estudiante: bitácora con el total de horas trabajadas por estudiante.
- Conclusiones
- Sugerencias y recomendaciones.
- Referencias

7. Entregables

- Código fuente con documentación interna.
- Documentación.
- Archivos necesarios para ejecutar el programa.

8. Evaluación

- Sincronización entre procesos 30 %
- Integración 20 %
- Estadísticas 15 %
- Modos 5 %
- Control de eventos y datos mostrados con elegancia 10 %
- Documentación 20 %

9. Fecha de entrega

- 19 de abril 19:00 por tecdigital.

10. Otros aspectos administrativos

- Para la revisión del proyecto se debe de entregar tanto la documentación como la implementación del software.
- No se reciben trabajos después de la hora indicada.
- En la revisión del proyecto pueden estar presentes el coordinador y asistente.
- Es responsabilidad del estudiante proveer los medios para poder revisar la funcionalidad del software, por ejemplo, si no se realiza la interfaz, se debe de proporcionar otro medio para la verificación, de lo contrario la nota será cero en los rubros correspondientes a la funcionalidad faltante.