TEC | Tecnológico de Costa Rica

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

Proyecto 1
Comunicación de procesos sincronizada

Fecha de asignación:	23 de Setiembre, 2025	Fecha de entrega:	16 de Octubre, 2025
Grupos:	2 personas	Profesor:	Jason Leitón Jiménez

1. Objetivo

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

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 de concurrencia y comunicación 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 y principios. 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 de alguna manera.

4. Descripción

La idea general de este proyecto es comunicar y sincronizar distintos procesos pesados a través de memoria compartida.

En el proyecto existen 4 tipos de procesos principales (*Heavy Process*). A continuación se detalla cada uno de ellos:

4.1. Inicializador

Este proceso es se encarga de establecer todas las variables compartidas, como contadores, punteros a memoria compartida, información de auditoría, semáforos, entre otros. Cuando este proceso se ejecuta en consola debe recibir al menos los siguientes datos:

- Identificador del espacio compartido.
- Cantidad de espacios para almacenar los valores ascii. Si recibe 10 como parámetro esto significa que solo hay espacio para 10 caracteres de manera simultánea.
- La llave de para desencriptar la información.

Además, es el encargado de indicar cuál es el archivo de donde se toman los caracteres a transferir.

Una vez que la memoria ha sido inicializada el proceso debe de terminar.

4.2. Emisor

Cuando se ejecuta este proceso debe de recibir 2 parámetros, que corresponden a los siguientes datos: modo de ejecución y la clave de 8 bits para codificar la información haciendo un xor entre cada caracter y el dato.

Este proceso llena cada espacio en memoria principal de manera **circular** con los caracteres que fueron cargados por el inicializador (deben ser codificados con la llave por medio de un xor), los cuales provienen de un archivo de texto. Cabe destacar que la memoria compartida debe ser lo más eficiente posible, esto significa que si el usuario coloca que la cantidad de caracteres en memoria serán 10, entonces solo se debe reservar el espacio que se ocupe para esos 10 datos, además, cada caracter que se requiera colocar debe tener como mínimo la siguiente información:

- Valor en ascii.
- Lugar en la estructura que fue introducido (índice).
- Hora que fue introducido.
- Cualquier otro dato que se necesite.

Es de suma importancia mencionar que no se permite el uso de **busy waiting** para controlar el llenado de memoria. Se sugiere tratar la memoria compartida como aritmética de punteros, con el fin de que el acceso sea similar al de un vector.

Cada vez que se introduzca un dato en memoria debe aparecer un print de forma elegante (diferencia de colores y en forma tabular alineada) la información antes mencionada. En caso de que no haya espacio para introducir más datos, el encodificador deberá bloquearse hasta que haya espacio (¡Cuidado con utilizar busy waiting!)

Debido a que la estructura estará en memoria compartida, se podrá tener n instancias de encodificadores al mismo tiempo, introduciendo valores de caracteres de forma ordenada, con el fin de que el decodificador tome los datos y los pueda reconstruir.

Nota: No se permite la sobre escritura de datos no leídos y todo el llenado de memoria compartida debe realizarse de manera **circular**.

4.3. Receptor

Cuando se ejecuta este proceso debe de recibir 2 parámetros, que corresponden a los siguientes datos: modo de ejecución y la clave para decodificar los caracteres. Cabe mencionar que se decodifica haciendo un xor entre la clave y el dato.

Este proceso será el encargado de leer los valores de la estructura en memoria compartida de manera **circular**, una vez extraído cada caracter se deberá de ver (en tiempo real) como el nuevo archivo se va formando poco a poco.

Cabe destacar que no se permite el uso de **busy waiting** para controlar la lectura de datos desde memoria, en caso de que el decodificador no tenga qué leer, se deberá de bloquear (¡Cuidado con utilizar busy waiting!), hasta que hayan datos no leídos.

Cada vez que se lea un dato deberá mostrar en consola de forma elegante (con colores diferentes y bien alineado), el caracter leido, fecha y hora que fue ingresado y la posición (índice) de donde fue leído. Se podrán tener n instancias de receptores captando los valores. Cada receptor debe tomar los datos de la misma memoria compartida.

Podrán existir n instancias de receptores extrayendo valores de memoria compartida para formar el texto. El texto extraído debe ser mostrado en consola de acuerdo con la lectura de cada letra.

4.4. Finalizador

El finalizador tendrá la responsabilidad de finalizar de manera elegante todos los procesos (No utilice kill). El objetivo de este proceso es que le mande una señal a los distintos procesos para que puedan terminar su ejecución de manera normal.

El proceso debe recibir una señal externa proveniente de algún mecanismo físico como por ejemplo un botón, switch entre otros, de tal manera que cuando se recibe la señal este sea capaz de avisar a todos los procesos involucrados para que puedan terminar sus ejecuciones.

El finalizador debe esperar a que todos los receptores y emisores se cierren para luego mostrar las estadísticas generales de una manera elegante (alineado y con distintos colores)

- Cantidad de caracteres transferidos:
- Cantidad de caracteres en memoria compartida.
- Cantidad de Emisores vivos y totales.

- Cantidad de receptores vivos y totales.
- Cantidad de memoria utilizada.

Es importante mencionar que las estadísticas aparecen una vez que se hayan cerrado todos los procesos.

4.5. Modo de ejecución

Los caracteres se pueden leer y escribir de dos maneras, las cuales se detallan a continuación.

- Automático: El emisor o receptor leerá o colocará los caracteres en memoria cada cierto tiempo definido por el usuario.
- Manual: El emisor o receptor leerá o colocará los caracteres cada vez que el usuario presione un enter.

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 para el control de escritura y lectura.

6. Documentación- Estilo IEEE-Trans (máximo 2 páginas)

Coloque pregunta y la respuesta de cada item.

- 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.

Para el atributo de Trabajo individual y en equipo se debe especificar 7 puntos (Se debe colocar pregunta y respuesta), los cuales son los siguientes:

- Indicar las estrategias para el trabajo individual y en equipo de forma equitativa e inclusiva en las etapas del proyecto (planificación, ejecución y evaluación).
- Indicar la planificación del trabajo mediante la identificación de roles, metas y reglas.
- Indicar cuales acciones promueven la colaboración entre los miembros del equipo durante el desarrollo del proyecto.
- Indicar cómo se ejecutan las estrategias planificadas para el logro de los objetivos.
- Indicar la evaluación para la el desempeño del trabajo individual y en equipo
- Indicar la evaluación para las estrategias utilizadas de equidad e inclusión.
- Indicar la evaluación para las acciones de colaboración entre los miembros del equipo

7. Entregables

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

8. Evaluación

- Algoritmo emisor y receptor 10 %
- Sincronización entre procesos 40 %
- Integración 10 %
- Estadísticas 10 %
- Control de eventos y datos mostrados con elegancia 10 %
- Documentación 20 %

9. Fecha de entrega

- 16 de Octubre 15:00 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.
- Cualquier ambigüedad o duda puede contactar al profesor para aclararla.
- En caso de que necesite algún requerimiento de hardware o software coméntelo con el profesor con anticipación.