SISTEMAS OPERATIVOS

ACTIVIDAD DE APRENDIZAJE 16
PROGRAMA 8

Profesor:
VIOLETA DEL ROCIO BECERRA VELAZQUEZ

VERDUZCO ROSALES LUIS ENRIQUE
223992388
Ingenieria en Computación

05/12/25

CUCEI DIVTIC D04

Objetivo de la Actividad

El objetivo principal de esta práctica fue extender la simulación de Paginación Simple (Round Robin) para incorporar el manejo de Procesos Suspendidos. Esto implicó simular el intercambio (swapping) de procesos entre la Memoria Principal y el almacenamiento secundario (Disco). Se debió implementar la lógica para que, mediante comandos de teclado, los procesos bloqueados pudieran ser enviados a disco (liberando sus marcos de memoria) y posteriormente recuperados a memoria principal cuando existiera espacio suficiente.

Notas sobre el Lenguaje y Herramientas

Para esta actividad continué utilizando tecnologías web (HTML, CSS y JavaScript) debido a su facilidad para crear interfaces gráficas dinámicas que permiten visualizar la memoria y las colas en tiempo real.

- HTML: Para la estructura de las nuevas tablas de suspendidos y modales.
- CSS: Para estilizar el estado de suspensión (diferenciación visual) y ajustar el layout para acomodar la nueva cola.
- JavaScript: Para toda la lógica de planificación, gestión de memoria (marcos y páginas) y manejo de eventos de teclado.

Lógica de Programación Implementada

La base del programa sigue siendo como en el programa 7 paginación, pero se añadieron las siguientes estructuras y validaciones para cumplir con los requerimientos de la Actividad:

1. Cola de Suspendidos:
   Se creó un nuevo arreglo (suspendidos = []) para almacenar los procesos que salen de la memoria RAM pero siguen vivos en el sistema (en disco).

2. Gestión de Memoria (Swap-Out - Tecla S):

   - Al presionar 'S', el algoritmo verifica si hay procesos en la cola de Bloqueados.
   - Si existen, se toma el primero, se cambia su estado a suspendido y se ejecuta una función crítica: liberarMemoria(proceso). Esta función recorre la Tabla de Páginas del proceso, marca esos marcos como 'Libres' en el mapa de memoria y los devuelve a la lista de marcosLibres.
   - Esto permite que otros procesos utilicen ese espacio inmediatamente.

3. Recuperación (Swap-In - Tecla R):

   - Al presionar 'R', se verifica el primer proceso en la cola de Suspendidos.
   - Se implementó una validación estricta: if (proc.paginasReq <= marcosLibres.length).

- Solo si hay suficientes marcos libres, el proceso regresa a memoria, se le asignan nuevos marcos (que pueden ser diferentes a los que tenía antes) y pasa a la cola de Listos.

4. **Condición de Parada:**

Tuve que modificar la condición de finalización del bucle principal (mainEjecucion). Ahora, el programa no termina si las colas de Listos o Bloqueados están vacías, sino que verifica explícitamente que suspendidos.length === 0.

Principales dificultades y sus soluciones

1. Sincronización de la Memoria al Suspender

- Dificultad:
  Al principio, al mover un proceso a suspendidos, visualmente desaparecía de la cola de bloqueados, pero sus colores permanecían pintados en la cuadrícula de memoria, lo que impedía que nuevos procesos entraran aunque teóricamente había espacio.

- Solución:
  Implementé la función liberarMemoria() que no solo borra la entrada en la tabla de páginas lógica, sino que iterativamente limpia los objetos memoria[i] (poniendo estado 'Libre') y actualiza el array marcosLibres. Esto aseguró que la fragmentación externa se gestionara correctamente al suspender.

2. Validación de Espacio al Regresar (Tecla R)

- Dificultad:
  Cuando intentaba regresar un proceso grande de disco a memoria, si no había espacio, el programa a veces sobreescribía memoria o lanzaba error.

- Solución:
  Antes de mover el proceso de la lista suspendidos, agregué una pre-verificación. Si paginasNecesarias > marcosLibres, el proceso se queda en disco y mostré una alerta visual temporal en el estado de la simulación indicando "No hay espacio suficiente", así solo regresa si hay espacio.

3. Persistencia de Datos en las Tablas

- Dificultad:
  El requerimiento pedía que en la tecla 'T' (Tabla de Páginas) y en el BCP se mostraran los datos correctos de los suspendidos. Inicialmente, al no tener marcos asignados, la tabla de páginas fallaba al intentar leer undefined.

- Solución:
  Modifiqué la función de renderizado de la tabla de páginas para detectar si el proceso está en la lista de suspendidos. Si es así, en lugar de intentar listar marcos

(que no tiene), imprime: En Disco (Swap), lo cual nos da mayor claridad visual sobre el estado del proceso.

Conclusión

La implementación de la suspensión de procesos me permitió comprender la importancia del *swapping* en los sistemas operativos modernos. Esta técnica es vital para aumentar el grado de multiprogramación, permitiendo que el sistema acepte más procesos de los que caben físicamente en la memoria RAM.

Programar la lógica de liberar y reasignar marcos dinámicamente reforzó mi entendimiento sobre la paginación: un proceso no necesita estar contiguo, ni siquiera necesita estar siempre en RAM, pero el sistema operativo debe llevar un control estricto (a través del PCB y tablas de páginas) de dónde está cada fragmento de información en cada momento. Esta práctica fue un excelente ejercicio de manejo de estructuras de datos y gestión de recursos compartidos.

Código fuente: https://github.com/UnluckyLuren/SisOperativos

Ejecutable: https://unluckyluren.github.io/SisOperativos/