UNIVERSIDAD DE LA CUENCA DEL PLATA

Autorizada Definitivamente por Decreto 091/2006 del Poder Ejecutivo Nacional

INGENIERÍA EN SISTEMAS DE INFORMACIÓN
Sistema operativo II

TRABAJO PRÁCTICO INTEGRADOR

Informe Teórico

Estudiante: Andrusyszyn Emiliano, Tarnowski Tobias, Ropat Agustin

Profesor: Ing. Juan de Dios Benitez; Lic. Itatí Malañuk

Materia: Sistemas operativos II

Año: Segundo

Año 2024

UNIVERSIDAD DE LA CUENCA DEL PLATA

Autorizada Definitivamente por Decreto 091/2006 del Poder Ejecutivo Nacional

Índice

Introducción

Introducción al Problema

En el desarrollo de sistemas operativos, la gestión de procesos y memoria es un aspecto crítico, especialmente en sistemas sin memoria virtual. En estos entornos, la administración se realiza directamente sobre la memoria física, lo que implica desafíos como la fragmentación y la necesidad de liberar manualmente la memoria ocupada por procesos inactivos o terminados. Además, en la gestión de procesos se deben considerar técnicas para el control de los recursos, como la exclusión mutua, evitando que procesos simultáneos accedan a los mismos recursos. Para cumplir con estas necesidades, se suelen utilizar estructuras de control como semáforos que permiten coordinar el acceso ordenado y seguro a estos recursos.

El simulador desarrollado busca ofrecer una visión práctica de estos conceptos, simulando tanto la asignación de memoria como el cambio de estados en los procesos. Al incorporar una representación visual, el usuario puede observar cómo los procesos avanzan y retroceden entre diferentes estados y cómo el sistema operativo asigna, libera y reorganiza la memoria de forma dinámica en respuesta a las necesidades del sistema.

Objetivos del Simulador

El propósito principal del simulador es replicar la dinámica de la gestión de procesos y de memoria en un sistema operativo sin memoria virtual. Los objetivos específicos del simulador incluyen:

1. **Visualización del ciclo de vida de los procesos**: Representar los estados de los procesos (Nuevo, Listo, Ejecutando, Bloqueado y Terminado) y su evolución.
2. **Simulación de la asignación de memoria**: Implementar y analizar algoritmos de asignación de memoria (First-Fit, Best-Fit, Worst-Fit y Next-Fit) y mostrar su impacto en la fragmentación y disponibilidad de la memoria.
3. **Gestión de recursos con semáforos**: Aplicar teoría de exclusión mutua mediante semáforos para gestionar la interrupción de procesos, simulando una espera activa y controlando el acceso a recursos compartidos.
4. **Utilización de memoria de intercambio (swap)**: Permitir el intercambio entre la memoria principal y la memoria de respaldo cuando el sistema alcanza su límite de capacidad en la RAM.

Desarrollo

Conceptos Técnicos Fundamentales

Gestión de Memoria sin Memoria Virtual

La **gestión de memoria sin memoria virtual** implica el uso directo de la memoria física, sin capas adicionales que simulen una mayor capacidad, como la paginación o la segmentación. En este sistema, cada proceso recibe un espacio fijo de memoria, lo cual es ideal para sistemas con recursos limitados y permite un control más sencillo. Sin embargo, esta técnica también trae desafíos, como la **fragmentación** y la **limitación en la cantidad de procesos simultáneos**.

Colas de Procesos y Solicitud de Recursos

Las **colas de procesos** son estructuras que el sistema operativo utiliza para organizar los procesos en distintos estados (Listo, Ejecutando, Bloqueado, etc.). Cuando un proceso necesita un recurso, envía una solicitud; si el recurso está disponible, el sistema se lo asigna. En caso contrario, el proceso espera en la cola de recursos hasta que el recurso esté libre. Este sistema ayuda a administrar eficientemente los recursos y optimizar el uso de la CPU.

Estados de los Procesos y Cómo Cambian

Los procesos en el sistema operativo pueden estar en distintos estados. A continuación se describe cada uno de estos estados y cómo se relacionan entre sí:

- **Nuevo (New)**: Representa el proceso en su fase inicial, donde se está creando y asignando la memoria necesaria.
- **Listo (Ready)**: En este estado, el proceso está esperando ser programado para usar la CPU, en una cola de espera (FIFO) junto a otros procesos.
- **Ejecutando (Running)**: El proceso ha obtenido acceso a la CPU y está en ejecución activa, utilizando los recursos asignados.
- **Bloqueado (Blocked)**: El proceso ha realizado una solicitud de recurso no disponible (por ejemplo, una operación de E/S) y queda en espera hasta que el recurso se libere.
- **Terminado (Terminated)**: El proceso ha completado su ejecución, liberando todos los recursos asignados.

El simulador emplea tiempos de simulación para los estados Bloqueado y Ejecutando, implementando pausas controladas que representan la espera del proceso en cada estado. Esto permite emular condiciones reales de espera y asignación de recursos.

Memoria Principal y Memoria de Intercambio (Swap)

La **memoria principal** es el espacio físico (RAM) en el que los procesos activos son cargados y ejecutados. Dado su tamaño limitado, la memoria principal en el simulador se establece con un límite predefinido y permite la asignación dinámica y la liberación de bloques de memoria.

La **memoria de intercambio o swap** representa una extensión de la memoria principal en dispositivos de almacenamiento secundarios, como un disco. Cuando la memoria principal está llena, el simulador mueve los procesos de baja prioridad o los que están en estado de Bloqueado a la memoria de intercambio, liberando espacio en la RAM. Este mecanismo permite que nuevos procesos ingresen en la memoria principal. Una vez que un proceso en la memoria de intercambio se encuentra en estado Listo, se vuelve a cargar en la memoria principal si hay espacio suficiente.

Algoritmos de Asignación de Memoria

Para la asignación de memoria, el simulador implementa los siguientes algoritmos, seleccionando el más adecuado según los requisitos del usuario:

1. **First-Fit**: Encuentra el primer bloque de memoria disponible que sea lo suficientemente grande para el proceso.
2. **Best-Fit**: Selecciona el bloque de memoria más pequeño que cumpla con la solicitud de memoria del proceso.
3. **Worst-Fit**: Asigna el bloque más grande disponible, dejando espacio suficiente para futuras asignaciones.

Algoritmos de Planificación de Procesos

Los procesos en la cola de Listo siguen diferentes algoritmos de planificación que determinan el orden en que son atendidos:

- **FIFO (First-In, First-Out)**: Los procesos se ejecutan en el orden en que ingresan en la cola de Listo.
- **Round-Robin (RR)**: Cada proceso recibe un quantum o fragmento de tiempo para ejecutarse antes de que se le pase el control al siguiente en la cola.
- **SJF (Shortest Job First)**: Los procesos con menor tiempo estimado de ejecución son los primeros en ejecutarse, reduciendo el tiempo promedio de espera.
- **Planificación con Prioridad**: Los procesos con mayor prioridad tienen preferencia, aunque esto puede derivar en problemas de inanición para aquellos con baja prioridad.

Implementación del Semáforo para Exclusión Mutua

Para evitar conflictos en el acceso a recursos compartidos y garantizar que solo un proceso tenga acceso a un recurso específico, se implementa un semáforo en el simulador. Esto es especialmente importante en el caso de interrupciones de procesos, donde un proceso en ejecución puede ser interrumpido si otro proceso necesita el mismo recurso o si el sistema necesita liberar memoria o CPU.

Funcionamiento del Semáforo en el Simulador

Un semáforo binario, inicializado en 1, controla el acceso a recursos como la CPU o los dispositivos de E/S. El semáforo sigue el siguiente esquema:

1. **Operación de Wait**: Cuando un proceso intenta acceder a un recurso protegido, verifica el valor del semáforo:
    - Si el semáforo es mayor que cero, decrece el valor en 1, permitiéndole al proceso acceder al recurso.
    - Si el semáforo es igual a cero, el proceso queda en espera en la cola de Bloqueado.
2. **Operación de Signal**: Una vez que el proceso libera el recurso, incrementa el semáforo, permitiendo que otro proceso acceda.

Desarrollo Práctico del Simulador

Diseño del Simulador

El simulador se construyó en Python utilizando la biblioteca Tkinter para la interfaz gráfica y random para simular la aleatoriedad de las solicitudes de memoria y cambios de estado.

1. **Interfaz gráfica**: Se utiliza Tkinter para visualizar el estado de la memoria y los procesos en tiempo real. Cada proceso se representa en un estado específico y se muestra el estado actual de la memoria, incluyendo los bloques ocupados y libres.
2. **Simulación del Ciclo de Vida de los Procesos**: Cada proceso cambia de estado aleatoriamente, siguiendo las probabilidades de transición configuradas (por ejemplo, un 33% de probabilidad de pasar de Ejecutando a Terminado).
3. **Memoria Swap**: Cuando no hay suficiente memoria en la RAM, el simulador traslada procesos a la memoria de intercambio, priorizando el retorno de estos procesos a la memoria principal en cuanto se libere espacio.

Implementación del Código

El código se estructura en varias clases clave:

1. **Clase Proceso:** Representa cada proceso individualmente, con atributos como ID, memoria requerida y estado actual.
2. **Clase Memoria:** Controla la asignación y liberación de memoria, implementando los algoritmos de asignación mencionados.
3. **Clase Simulador:** Se encarga de la lógica del simulador, incluyendo la creación de procesos, la transición de estados y la actualización de la interfaz.

Resultados

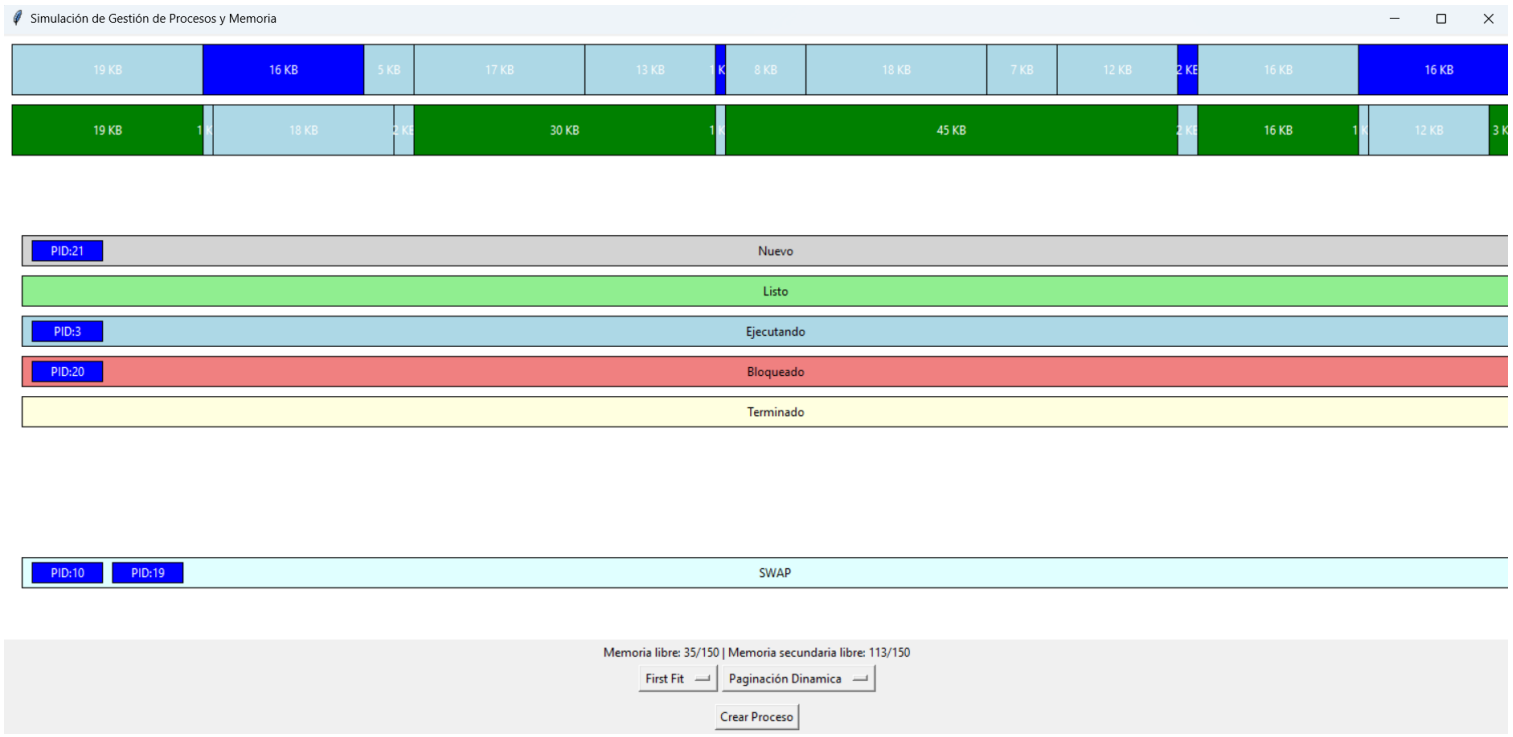
Al ejecutar el simulador, observamos los siguientes comportamientos:

Asignación y Fragmentación de Memoria: Con First-Fit, la memoria se asigna rápidamente, pero genera fragmentación en el tiempo, lo que se alinea con los problemas teóricos de fragmentación interna y externa. Best-Fit y Worst-Fit tienen tiempos de búsqueda más largos, pero en ciertos escenarios, logran una mejor utilización de la memoria.

Estados de Procesos y Semáforos: Implementamos semáforos para simular interrupciones y la gestión de recursos en la transición entre estados. Los procesos pueden ser interrumpidos aleatoriamente para simular eventos externos, como solicitudes de entrada/salida. Estos eventos se gestionan mediante semáforos, permitiendo una administración segura y sincronizada del uso de recursos.

Eficiencia del Simulador: Medimos el impacto de diferentes algoritmos en tiempos de ejecución y gestión de memoria. Round-Robin, con un quantum ajustado, se demostró como una técnica eficiente para distribuir el tiempo de CPU en sistemas interactivos, minimizando el riesgo de inanición.

Imagen:


Simulación de Gestión de Procesos y Memoria
19 KB
16 KB
5 KB
17 KB
13 KB
8 KB
18 KB
7 KB
12 KB
16 KB
16 KB
19 KB
18 KB
30 KB
45 KB
16 KB
12 KB
PID:21
Nuevo
Listo
PID:3
Ejecutando
PID:20
Bloqueado
Terminado
PID:10
PID:19
SWAP
Memoria libre: 35/150 | Memoria secundaria libre: 113/150
First Fit
Paginación Dinamica
Crear Proceso

Conclusión

El desarrollo del simulador nos permitió comprender en profundidad los desafíos y las soluciones en la gestión de memoria y procesos sin memoria virtual. A través de esta experiencia, identificamos las limitaciones y ventajas de los distintos algoritmos de asignación y planificación. La creación de una interfaz gráfica facilitó la visualización de los procesos y su interacción con la memoria, destacando la importancia de un diseño bien estructurado para implementar simulaciones de sistemas complejos.