



Samuel Padilla Belvis

Práctica 3

Práctica 3 de Sistemas Operativos

Fecha límite de entrega 22/12/2023

Descripción General

Este script en Python implementa una simulación de gestión de memoria para sistemas operativos, utilizando algoritmos de asignación de memoria dinámica.

Clases y Métodos

Clase Process

Proporciona funciones de entrada/salida, como printf para imprimir en la consola y fopen, fgets, fclose para manejar archivos.

- **Descripción:**

Representa un proceso con atributos como nombre, tiempo de llegada, memoria requerida y tiempo de ejecución.

- **Atributos:**

- name: Nombre del proceso.
- arrival_time: Instante de llegada del proceso.
- memory_required: Cantidad de memoria requerida por el proceso.
- execution_time: Tiempo de ejecución del proceso.
- end_time: Tiempo en el que el proceso termina su ejecución.

Clase MemoryPartition

- **Descripción:**

Representa una partición de memoria.

- **Atributos:**

- start: Dirección de inicio de la partición.
- size: Tamaño de la partición.
- process: Proceso asignado a la partición (si existe).

- **Métodos:**

- __str__(): Devuelve una representación en cadena de la partición de memoria.

Clase MemoryManager

- **Descripción:**

Gestiona las particiones de memoria y asigna memoria a los procesos.

- **Atributos:**

- total_memory: Tamaño total de la memoria disponible.
- partitions: Lista de particiones de memoria.

- **Métodos:**

- best_fit(process): Asigna memoria al proceso utilizando el algoritmo de mejor ajuste (Best Fit).
- worst_fit(process): Asigna memoria al proceso utilizando el algoritmo de peor ajuste (Worst Fit).
- allocate_memory(process, allocation_algorithm): Asigna memoria al proceso según el algoritmo especificado.
- free_memory(current_time): Libera la memoria de los procesos que han terminado su ejecución.
- merge_free_partitions(): Fusiona particiones de memoria contiguas que están libres.
- run_simulation(processes, allocation_algorithm): Ejecuta la simulación de asignación de memoria.
- get_memory_state(current_time): Devuelve el estado actual de la memoria y su representación visual.

Funciones Auxiliares

`read_input_file(file_path)`

- **Descripción:**

Lee un archivo de entrada y crea una lista de procesos.

- **Parámetros:**

- `file_path`: Ruta del archivo de entrada.

`write_output_file(file_path, memory_states)`

- **Descripción:**

Escribe el estado de la memoria en un archivo de salida.

- **Parámetros:**

- `file_path`: Ruta del archivo de salida.
- `memory_states`: Estados de la memoria para escribir en el archivo.

`main(input_file, output_file)`

- **Descripción:**

Función principal que ejecuta la simulación de gestión de memoria.

- **Parámetros:**

- `input_file`: Archivo de entrada con información de procesos.
- `output_file`: Archivo de salida para guardar el estado de la memoria.

Ejecución

El script se ejecuta desde la línea de comandos con dos argumentos: el archivo de entrada y el archivo de salida.

Ejemplo: `python main.py entrada.txt particiones.txt`