

Reporte Técnico

Portada

Título: Simulador de Gestor de Memoria RAM y Swap
Equipo: [Nombres de los integrantes]
Fecha: 2 de diciembre de 2025

Resumen

Este documento describe la implementación de un simulador del gestor de memoria que soporta multiprogramación, paginación, swapping y visualización del estado de la memoria. El proyecto forma parte de un ejercicio académico cuyo objetivo es comprender el funcionamiento interno de un sistema operativo y está desarrollado en Python con una interfaz de línea de comandos.

1. Descripción general del simulador

El simulador recibe como entrada procesos con requisitos de memoria específicos y asigna marcos en RAM y en Swap utilizando un esquema de paginación. Cuando la memoria física se llena, emplea un algoritmo de reemplazo FIFO (First-In, First-Out) para seleccionar páginas víctimas y realizar el intercambio hacia el área de Swap. El usuario puede visualizar el mapa de memoria, consultar la tabla de páginas de cada proceso, acceder a páginas específicas y conocer métricas de rendimiento del sistema 【823107480250600†L73-L99】 .

2. Módulos principales

Módulo	Descripción
config.py	Lee el archivo config.ini para obtener el tamaño de la RAM, el tamaño del área de Swap y el tamaño de página; calcula el número de marcos totales disponibles en cada memoria 【823107480250600†L50-L58】 .
process.py	Define la clase Process, que incluye el identificador único del proceso (PID), su tamaño en kilobytes, el número de páginas necesarias y una tabla de páginas con bits de presencia y referencias.
memory_manager.py	Implementa el gestor de memoria. Mantiene las listas de marcos de RAM y Swap, asigna páginas a marcos, implementa el algoritmo FIFO para reemplazar páginas cuando la RAM está llena, gestiona el swapping y mantiene métricas.
replacement.py	Contiene la clase abstracta ReplacementAlgorithm y la implementación concreta FIFOReplacement. Esta

Módulo	Descripción
	arquitectura facilita la incorporación de otros algoritmos como LRU o reloj 【823107480250600†L88-L93】 .
cli.py	Proporciona la interfaz de línea de comandos que permite al usuario crear procesos, terminarlos, acceder a páginas, visualizar el mapa de memoria y consultar métricas.
logger.py	Registra los eventos generados durante la ejecución: creación y terminación de procesos, fallos de página y movimientos de páginas entre RAM y Swap.

3. Ejemplo de ejecución

Se muestran a continuación algunos ejemplos de interacción con el simulador para ilustrar su funcionamiento:

- **Creación de procesos:**
 - *Entrada:* Tamaño del proceso = 512 KB – *Salida:* Proceso 1 creado con tamaño 512 KB y 2 páginas.
 - *Entrada:* Tamaño del proceso = 768 KB – *Salida:* Proceso 2 creado con tamaño 768 KB y 3 páginas. Página 0 del Proceso 1 movida a Swap en marco 0 por falta de espacio.
- **Acceso a página:**
 - *Entrada:* PID = 2, página = 2 – *Salida:* Acceso exitoso a P2 página 2 en RAM[3] (TLB hit).
- **Métricas de rendimiento al final de la ejecución:**
 - Fallas de página: 1
 - Operaciones de swapping: 1
 - Marcos usados en RAM: 5/8 (62,5 % de utilización)

4. Análisis del algoritmo de reemplazo

El algoritmo FIFO selecciona como víctima la página que ingresó primero a la RAM. Su principal ventaja radica en su simplicidad: se implementa mediante una cola y no requiere información adicional. Sin embargo, puede elegir páginas que se utilizan con frecuencia, produciendo un mayor número de fallos de página en comparación con estrategias que consideran la temporalidad de acceso, como **LRU** (Least Recently Used) o el **algoritmo del reloj**. En futuras versiones se podría implementar LRU para priorizar la retención de páginas recientemente utilizadas, aunque con un mayor costo de mantenimiento de las estructuras de datos.

5. Conclusiones

El simulador desarrollado cumple con los requisitos establecidos: permite visualizar el estado de la memoria física y del área de Swap, gestionar la llegada y salida de procesos en un entorno de multiprogramación, implementar una política de reemplazo de páginas y proporcionar métricas de rendimiento 【823107480250600†L73-L99】. Su arquitectura modular facilita la extensión a otros algoritmos de reemplazo y la integración de interfaces gráficas en el futuro.