Universidad Autónoma de TAMAULIPAS

Facultad de Ingeniería Tampico

Proyecto Integrador

Sistemas Operativos

Manual de Usuario

Integrantes de Equipo:

Kim Martínez Taesoo

Guzmán García Lizbeth Neri

Chavarría Uribe Leo

Vicente Bautista Roberto Nicolas

Zertuche Zamora Sergio Gabriel

Periodo 2025-3.

Introducción

El Simulador de Gestor de Memoria RAM (SGMR) es una aplicación en consola desarrollada en Python que permite observar de forma práctica cómo un sistema operativo administra la memoria principal y la memoria de intercambio utilizando paginación y un algoritmo de reemplazo de páginas FIFO.
A través de un menú interactivo, el usuario puede crear y terminar procesos, revisar el estado de la memoria, consultar tablas de páginas y obtener estadísticas globales del sistema. Este manual explica, paso a paso, cómo preparar el entorno, configurar el simulador y utilizar cada una de sus opciones.

Objetivo del manual

El objetivo de este manual es guiar al usuario en el uso correcto del simulador SGMR, desde la configuración inicial hasta la ejecución de pruebas, de manera clara y sencilla. No es necesario comprender todo el código fuente; basta con seguir las indicaciones descritas en este documento.

Requisitos previos

Antes de utilizar el simulador, asegúrate de contar con lo siguiente:

- Python 3.8 o superior instalado en tu equipo.
- Un editor o IDE de tu preferencia (por ejemplo, VS Code, PyCharm, Thonny o similar).
- Acceso a una terminal o consola (CMD, PowerShell, Terminal de Linux/macOS).

El simulador puede ejecutarse en Windows, Linux o macOS, siempre y cuando Python esté correctamente instalado y puedas abrir una consola para ejecutar los comandos.

Estructura básica del proyecto

Tras descomprimir el archivo del proyecto, encontrarás una estructura similar a la siguiente (puede variar ligeramente):

- ui.py – Interfaz de usuario en consola.
- simulator.py – Lógica principal del simulador de sistema operativo.
- memory.py – Administración de marcos de memoria RAM y SWAP.
- process.py – Representación de procesos y sus tablas de páginas.
- paging.py – Sistema de paginación y manejo de fallos de página.
- config_loader.py – Carga y validación del archivo de configuración.
- config.ini – Archivo donde se definen los parámetros de memoria.
- test.py – Conjunto de pruebas automáticas del simulador.
- docs/test_logs – Archivo de texto con los resultados de las pruebas.

Para la ejecución normal como usuario, los archivos más importantes son ui.py, config.ini y, de forma opcional, test.py.

Configuración inicial

Antes de ejecutar el simulador, es recomendable revisar el archivo config.ini, ya que define el tamaño de la memoria que se simulará.
Un ejemplo de configuración es el siguiente:

```
[MEMORY]
ram_size = 2048
swap_size = 4096
page_size = 256
```

Los parámetros tienen el siguiente significado:

- ram_size: tamaño total de la memoria física simulada (RAM), en bytes.
- swap_size: tamaño total del área de intercambio (SWAP), en bytes.
- page_size: tamaño de cada página y cada marco de memoria, en bytes.

El simulador utiliza estos valores para calcular automáticamente el número de marcos disponibles en RAM y en SWAP. Por ejemplo, con la configuración anterior:

- Marcos en RAM = ram_size / page_size = 2048 / 256 = 8 marcos.
- Marcos en SWAP = swap_size / page_size = 4096 / 256 = 16 marcos.

Es posible modificar estos valores para simular diferentes escenarios (por ejemplo, más RAM o más SWAP). Se recomienda usar siempre números enteros y mantener un tamaño de página razonable para que los resultados sean fáciles de interpretar.

Preparación del entorno

Sigue estos pasos para preparar el entorno de ejecución del simulador:

- Descomprime el archivo .zip del proyecto en una carpeta de tu preferencia.
- Abre una terminal o consola en tu sistema operativo.
- Navega hasta la carpeta donde se encuentran los archivos fuente (por ejemplo, la carpeta src).
- Verifica que el archivo ui.py esté presente en esa carpeta.

Opcionalmente, puedes crear un entorno virtual de Python para aislar las dependencias del proyecto, aunque el simulador está pensado para funcionar únicamente con la biblioteca estándar de Python.

Ejecución del simulador

Para iniciar el simulador desde la terminal, asegúrate de estar en la carpeta donde se encuentra ui.py y ejecuta:

python ui.py

En algunos sistemas, especialmente en Linux o macOS, puede ser necesario utilizar:

python3 ui.py

Si todo está configurado correctamente, aparecerá en pantalla el menú principal del simulador.

Menú principal y opciones

El menú principal presenta un conjunto de opciones numeradas que permiten interactuar con el simulador. Aunque el formato exacto puede variar, generalmente se muestra de la siguiente manera:

```
========================================
      SIMULADOR DE MEMORIA (FIFO)
========================================
1. Crear nuevo proceso
2. Terminar proceso
3. Ver estado de RAM y SWAP
4. Ver tabla de páginas de un proceso
5. Ver estadísticas globales
6. Salir
----------------------------------------
Selecciona una opción:
```

A continuación, se describen las funciones de cada opción.

Opción 1 – Crear nuevo proceso

Al seleccionar esta opción, el simulador solicita el tamaño del proceso en bytes. Con ese dato, calcula cuántas páginas serán necesarias y trata de asignarlas a marcos de memoria en la RAM. Si la memoria principal está llena, entra en juego el algoritmo de reemplazo FIFO, que mueve páginas antiguas a la memoria de intercambio (SWAP) para liberar espacio.
Ejemplo: si el tamaño de página es de 256 bytes y se crea un proceso de 600 bytes, el simulador calculará que se requieren 3 páginas (redondeando hacia arriba). Cada una de ellas será ubicada en algún marco de RAM o, si fuera necesario, en SWAP.

Opción 2 – Terminar proceso

Esta opción permite finalizar un proceso existente. El simulador solicitará el ID del proceso que se desea terminar y, si es válido, liberará todos los marcos de RAM y de SWAP ocupados por las páginas de dicho proceso. Esto deja la memoria disponible para nuevos procesos.

Opción 3 – Ver estado de RAM y SWAP

Muestra el contenido actual de los marcos en RAM y en SWAP. Los marcos libres suelen aparecer marcados como LIBRE, mientras que los marcos ocupados muestran el ID del proceso y el número de página que están almacenando.
Esta vista es útil para entender cómo se distribuyen las páginas de los procesos en la memoria física y en el área de intercambio.

Opción 4 – Ver tabla de páginas de un proceso

Permite consultar la tabla de páginas de un proceso en particular. Para ello, el simulador pide el ID del proceso y muestra, para cada una de sus páginas lógicas, si se encuentra en RAM o en SWAP y el marco correspondiente.
Este apartado es especialmente útil para relacionar el concepto teórico de tabla de páginas con una visualización concreta dentro del simulador.

Opción 5 – Ver estadísticas globales

Presenta un resumen del estado general del sistema de memoria, que puede incluir, entre otros datos:

- Número de marcos libres y totales en RAM.
- Uso actual de marcos en SWAP.
- Conteo acumulado de fallos de página.

Estas estadísticas permiten evaluar el comportamiento del sistema bajo diferentes cargas de trabajo.

Opción 6 – Salir del simulador

Cierra de forma ordenada la aplicación y regresa el control a la terminal. Se recomienda utilizar esta opción para finalizar el programa en lugar de cerrar la ventana de la consola abruptamente.

Casos de uso sugeridos

A continuación, se proponen algunos escenarios para explorar el funcionamiento del simulador:

- Caso A – Prueba básica: crear dos o tres procesos pequeños y observar cómo se asignan sus páginas en RAM.
- Caso B – Forzar uso de SWAP: crear procesos de gran tamaño hasta llenar la RAM y revisar qué páginas son enviadas al área de intercambio.
- Caso C – Limpieza de memoria: después de crear varios procesos, terminar uno de ellos y verificar que sus marcos se liberen correctamente en RAM y en SWAP.

Pruebas automáticas

El archivo test.py contiene una serie de pruebas automatizadas que ejercitan diferentes partes del simulador.
Para ejecutarlas, abre la terminal en la carpeta del proyecto y escribe:
python test.py
Dependiendo de tu sistema, también puedes utilizar
python3 test.py
Durante la ejecución se mostrarán en consola los diferentes escenarios de prueba y un resumen de los resultados quedará registrado en docs/test_logs.
Entre las pruebas habituales se incluyen casos de carga básica de procesos, llenado de la memoria para forzar el swapping, verificación de la integridad de las tablas de páginas y terminación correcta de procesos.

Problemas frecuentes y soluciones

A continuación, se describen algunos errores comunes y cómo resolverlos:

- El archivo config.ini no existe o no se encuentra.

Verifica que config.ini esté en la misma carpeta desde la cual se ejecuta ui.py. En caso de que falte, crea uno nuevo utilizando el ejemplo de la sección de configuración inicial.

- El simulador indica que no se puede leer config.ini.
  Revisa que el archivo tenga la sección [MEMORY] y que los parámetros estén bien escritos, sin caracteres extraños ni comentarios mal colocados dentro de las líneas de configuración.
- El comando python o python3 no se reconoce.
  En Windows, asegúrate de haber agregado Python al PATH durante la instalación. Si el problema persiste, puedes volver a instalar Python seleccionando la opción correspondiente.
- Se introduce texto en lugar de números al crear o terminar procesos.
  El simulador espera valores numéricos enteros para el tamaño del proceso y para los IDs. Si se introduce texto o símbolos, pueden aparecer errores o mensajes indicando que el valor es inválido.

Recomendaciones finales

El simulador SGMR está diseñado como herramienta didáctica para apoyar el aprendizaje de la gestión de memoria en sistemas operativos. Se recomienda experimentar con distintos tamaños de procesos y configuraciones de memoria para observar cómo cambia el comportamiento del sistema.
Si solamente vas a utilizar el simulador como usuario, no es necesario modificar el código fuente. En la mayoría de los casos bastará con ajustar, si se desea, los valores del archivo config.ini y ejecutar la interfaz por consola.
Finalmente, se sugiere complementar el uso del simulador con la teoría vista en clase sobre paginación, marcos, fallos de página y memoria de intercambio, para aprovechar al máximo esta herramienta.