

Proyecto Integrador

Sistemas Operativos

Manual Técnico

UNIVERSIDAD AUTÓNOMA DE TAMAULIPAS

Integrantes de Equipo

Kim Martínez Taesoo

Guzmán García Lizbeth Neri

Chavarría Uribe Leo

Vicente Bautista Roberto Nicolas

Zertuche Zamora Sergio Gabriel

Módulos

Módulo 1 — config_loader.py

Carga y validación de configuración

Función principal

Leer, validar y procesar el archivo config.ini, asegurando que el simulador arranque con parámetros correctos.

Responsabilidades técnicas

- Verificar que el archivo *config.ini* exista y pueda ser leído.
- Validar que contenga la sección obligatoria [MEMORY].
- Validar las claves:
 - o ram_size
 - o swap_size
 - o page_size
- Convertir los valores a enteros y revisar que sean positivos.
- Validar que:
 - o $\text{ram_size \% page_size} == 0$
 - o $\text{swap_size \% page_size} == 0$
- Calcular:
 - o frames_ram
 - o frames_swap

Aspectos importantes

- ! Se utiliza configparser.ConfigParser() para lectura estructurada.
- ! Se emplean salidas controladas con sys.exit(1) en errores críticos.
- ! Retorna un diccionario con toda la configuración procesada.

Módulo 2 — process.py

Representación interna del proceso y tabla de páginas

Función principal

Modelar un proceso como una entidad que posee una tabla de páginas y estado.

Responsabilidades técnicas

- Calcular el número de páginas necesarias basándose en el tamaño del proceso.
- Crear la tabla de páginas como un diccionario estructurado.
- Administrar el estado del proceso:
 - o new: nuevo proceso
 - o active: proceso activo
 - o partially_swapped: proceso parcialmente intercambiado
 - o swapped: proceso intercambiado
 - o terminated: proceso terminado
- Permitir la actualización de:
 - o frame asignado
 - o ubicación (RAM o SWAP)
 - o bit de presencia

Estructura usada

```
self.page_table = {  
    page_number: {  
        'frame_id': int | None,  
        'location': 'RAM' | 'SWAP' | None,  
        'present': bool  
    }  
}
```

Aspectos importantes

- ! Se usa `math.ceil()` para el cálculo de páginas.
- ! La clase no gestiona memoria directamente; solo refleja el estado del proceso.
- ! La tabla es impresa en formato estructurado para facilitar el análisis.

Módulo 3 — memory.py

Gestión de RAM, SWAP y algoritmo FIFO

Función principal

Simular la memoria física y el área de intercambio (SWAP) mediante objetos Frame.

3.1 — Clase Frame

Representa un marco individual de memoria.

Atributos

- `id`
- `process_id`
- `page_number`
- `free` (True/False)

Métodos clave

- `assign(pid, page)`
- `free_form`

3.2 — Clase MemoryManager

Responsabilidades técnicas

- Mantener listas:
 - o `ram = [Frame]`
 - o `swap = [Frame]`
- Buscar marcos libres de manera secuencial.
- Asignar y liberar marcos.
- Implementar el algoritmo FIFO con una cola deque.

Estructuras utilizadas

RAM y SWAP

```
self.ram = [Frame(i) for i in range(frames_ram)]  
self.swap = [Frame(i) for i in range(frames_swap)]
```

FIFO

```
self fifo_queue = deque()
```

Funciones críticas

- `allocate_ram(pid, page)`
- `free_ram_frame(frame_id)`
- `select_victim_fifo()`
- `move_ram_to_swap(frame_id)`
- `get_free_ram_frame()`
- `get_free_swap_frame()`

Aspectos importantes

- ! FIFO selecciona al marco más antiguo que fue asignado.
- ! SWAP se llena secuencialmente, simulando discos reales.
- ! El módulo NO decide sobre procesos; solo mueve marcos.

Módulo 4 — paging.py

Sistema de paginación, swapping y fallos de página

Función principal

Gestionar la memoria virtual y decidir cómo entran y salen páginas de RAM/SWAP.

Responsabilidades principales

- Cargar procesos página por página.
- Detectar falta de espacio en RAM.
- Activar swapping usando FIFO.
- Gestionar page faults.
- Sincronizar la tabla de páginas del proceso.
- Ejecutar swap-in a demanda.

Funciones importantes

load_process(process)

Intenta asignar cada página a RAM. Si RAM está llena:

- aumenta contador de page_faults
- selecciona víctima vía FIFO
- mueve víctima a SWAP
- actualiza tabla de páginas de víctima
- asigna marco liberado al proceso actual

swap_in(pid, page)

Para cuando una página en SWAP necesita RAM, puede causar otro page fault y usa FIFO para desalojar otra página si es necesario.

unload_process(pid)

Libera RAM y SWAP del proceso y marca el proceso como terminated.

Estructuras utilizadas

Procesos activos

```
self.active_processes = {}
```

Page faults

```
self.page_faults = 0
```

Módulo 5 — simulator.py

Capa integradora del sistema operativo

Función principal

Coordinar todos los módulos como si fuera un kernel simplificado.

Responsabilidades técnicas

- Instanciar:
 - o Configuración
 - o MemoryManager
 - o PagingSystem
- Gestionar creación y eliminación de procesos
- Proveer estadísticas globales
- Exponer métodos para la interfaz del usuario

Métodos clave

- create_process(size)
- terminate_process(pid)
- show_memory_status()
- show_page_table(pid)
- get_global_stats()

Estructura importante

```
self.processes = { pid : Process }
```

Módulo 6 — ui.py

Interfaz de usuario tipo CLI

Función principal

Proveer una interfaz interactiva para operar el simulador.

Características técnicas

- **Menú persistente con opciones:**
 - Crear proceso
 - Terminar proceso
 - Ver RAM/SWAP
 - Ver tabla de páginas
 - Ver estadísticas globales
 - Salir
- Limpieza de pantalla por sistema operativo.
- Validación de entradas numéricas.
- Uso directo de los métodos expuestos por OSSimulator.

Aspectos importantes

- ! Permite demostrar el funcionamiento del simulador en tiempo real.
- ! No implementa lógica interna; solo llama a los módulos correctos.

Casos de prueba

El archivo test.py implementa un conjunto de pruebas automatizadas de integración para validar el correcto funcionamiento del simulador de memoria.

Estas pruebas no evalúan componentes aislados, sino el flujo completo: creación de procesos, asignación de memoria, swapping, tablas de página y finalización.

Objetivo del módulo de pruebas

El propósito del módulo es verificar:

- Que la memoria RAM y SWAP se gestionen correctamente.
- Que el algoritmo FIFO seleccione víctimas apropiadamente.
- Que el contador de fallos de página (page_faults) aumente cuando corresponde.
- Que las tablas de páginas de los procesos se mantengan coherentes.
- Que el simulador se mantenga estable ante distintas cargas.
- Que el proceso de terminación libere correctamente todos los marcos.

Es decir, asegura que el comportamiento del simulador sea determinista, consistente y correcto, incluso bajo condiciones de presión de memoria.

Etapas

Test 1 — test_1_basic_process_load()

Objetivo

Validar que el simulador cargue procesos en RAM cuando existe suficiente espacio, sin activar swapping.

Acciones

- Crea dos procesos pequeños.
- Verifica que:
 - RAM se actualiza correctamente.
 - No hay page faults.
 - Ninguna página va a swap.

Relevancia

Asegura que la carga base de procesos (sin presión de memoria) funcione correctamente.

Test 2 — test_2_force_swapping()

Objetivo

Verificar el funcionamiento del algoritmo de reemplazo FIFO bajo condiciones de memoria llena.

Acciones

- Crea un proceso grande que excede la capacidad de RAM.
- Obliga al sistema a realizar swapping.
- Verifica que:
 - $\text{page_faults} > 0$
 - FIFO haya ejecutado correctamente el reemplazo.
 - Algunos marcos terminen en SWAP.

Relevancia

Prueba la parte más crítica del simulador: el mecanismo de swapping y reemplazo de páginas.

Test 3 — test_3_page_table_integrity()

Objetivo

Validar que la tabla de páginas de un proceso se construya correctamente.

Acciones

- Crea un proceso y muestra su tabla de páginas.
- Verifica:
 - Que el número de páginas coincida con lo calculado.
 - Que el estado del proceso sea coherente (active / partially_swapped).
 - Que no haya entradas corruptas.

Relevancia

Garantiza que la estructura fundamental del sistema de paginación funcione: la tabla de páginas.

Test 4 — test_4_swap_in_operation()

Objetivo

Validar la operación swap-in, es decir, traer una página desde SWAP a RAM.

Acciones

- Crea un proceso grande que genera páginas en SWAP.
- Localiza una página del proceso que esté en SWAP.
- Llama explícitamente a swap_in(pid, page_number).
- Verifica que:
 - La página retorna a RAM.
 - Se libera correctamente el marco en SWAP.
 - La tabla de páginas del proceso se actualiza.

Relevancia

Prueba una operación avanzada: la recuperación de páginas desde disco a RAM, simulada correctamente.

Test 5 — test_5_process_termination()

Objetivo

Validar la correcta liberación de RAM y SWAP cuando un proceso termina.

Acciones

- Crea dos procesos.
- Termina el proceso 1.
- Verifica:
 - Que se eliminó de sim.processes.
 - Que sus marcos en RAM fueron liberados.
 - Que sus marcos en SWAP fueron liberados.
 - Que cambia su estado a terminated.

Relevancia

Garantiza la integridad del sistema tras la terminación de procesos: no deben quedar marcos sucios ni referencias colgantes.

Ejemplo de ejecución

test.py

```
kim@kims-MacBook-Air src % python3 test.py

=====
TEST: INICIANDO PRUEBAS DEL SISTEMA OPERATIVO
=====

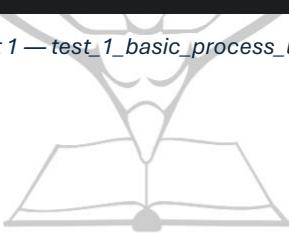
=====
TEST: 1) Carga básica de procesos
=====

[Simulador] Creando Proceso 1 (200 bytes)...
[Simulador] Requiere 1 páginas.
[Paging] Cargando proceso 1 (1 pags)...
[Paging] Proceso 1 listo.
[Exito] Proceso 1 cargado en memoria.

[Simulador] Creando Proceso 2 (300 bytes)...
[Simulador] Requiere 2 páginas.
[Paging] Cargando proceso 2 (2 pags)...
[Paging] Proceso 2 listo.
[Exito] Proceso 2 cargado en memoria.

Estadísticas: {'Marcos RAM Libres': '5 / 8', 'Uso de Swap': '0 / 16', 'Fallos de Página': 0}
```

Test 1 — test_1_basic_process_load



```
=====
TEST: 2) Forzar swapping llenando RAM
=====

[Simulador] Creando Proceso 1 (3000 bytes)...
[Simulador] Requiere 12 páginas.
[Paging] Cargando proceso 1 (12 pags)...
  [!] RAM llena (Pag 8). Ejecutando FIFO...
    > SwapOut: Marco 0 (P1, Pag 0) -> Swap 0
  [!] RAM llena (Pag 9). Ejecutando FIFO...
    > SwapOut: Marco 1 (P1, Pag 1) -> Swap 1
  [!] RAM llena (Pag 10). Ejecutando FIFO...
    > SwapOut: Marco 2 (P1, Pag 2) -> Swap 2
  [!] RAM llena (Pag 11). Ejecutando FIFO...
    > SwapOut: Marco 3 (P1, Pag 3) -> Swap 3
[Paging] Proceso 1 listo.
[Exito] Proceso 1 cargado en memoria.
```

Ver estado final de RAM:

```
--- ESTADO DE RAM---
[Frame 0: P1, Pag 8]
[Frame 1: P1, Pag 9]
[Frame 2: P1, Pag 10]
[Frame 3: P1, Pag 11]
[Frame 4: P1, Pag 4]
[Frame 5: P1, Pag 5]
[Frame 6: P1, Pag 6]
[Frame 7: P1, Pag 7]
```

```
--- ESTADO DE SWAP---
[Frame 0: P1, Pag 0]
[Frame 1: P1, Pag 1]
[Frame 2: P1, Pag 2]
[Frame 3: P1, Pag 3]
[Frame 4: LIBRE]
[Frame 5: LIBRE]
[Frame 6: LIBRE]
[Frame 7: LIBRE]
[Frame 8: LIBRE]
[Frame 9: LIBRE]
[Frame 10: LIBRE]
[Frame 11: LIBRE]
[Frame 12: LIBRE]
[Frame 13: LIBRE]
[Frame 14: LIBRE]
[Frame 15: LIBRE]
```

Estadísticas: {'Marcos RAM Libres': '0 / 8', 'Uso de Swap': '4 / 16', 'Fallos de Página': 4}

Test 2 — test_2_force_swapping

```
=====
TEST: 3) Integridad de tabla de páginas
=====

[Simulador] Creando Proceso 1 (700 bytes)...
[Simulador] Requiere 3 páginas.
[Paging] Cargando proceso 1 (3 pags)...
[Paging] Proceso 1 listo.
[Exito] Proceso 1 cargado en memoria.

Tabla de páginas del proceso:

--- Tabla de Paginas: Proceso ID 1 ---
Tamaño: 700 bytes | Total Paginas: 3
-----
Pagina | Ubicacion | Marco Fisico | En RAM
-----
0     | RAM       | 0           | Si
1     | RAM       | 1           | Si
2     | RAM       | 2           | Si
```

Test 3 — test_3_page_table_integrity

```
=====
TEST: 4) Test explícito de swap-in
=====

[Simulador] Creando Proceso 1 (1500 bytes)...
[Simulador] Requiere 6 páginas.
[Paging] Cargando proceso 1 (6 pags)...
[Paging] Proceso 1 listo.
[Exito] Proceso 1 cargado en memoria.
No hubo páginas en swap. Cambiar tamaños para test.
```

Test 4 — test_4_swap_in_operation

```
=====
TEST: 5) Terminación de proceso y limpieza
=====

[Simulador] Creando Proceso 1 (900 bytes)...
[Simulador] Requiere 4 páginas.
[Paging] Cargando proceso 1 (4 pags)...
[Paging] Proceso 1 listo.
[Exito] Proceso 1 cargado en memoria.

[Simulador] Creando Proceso 2 (1200 bytes)...
[Simulador] Requiere 5 páginas.
[Paging] Cargando proceso 2 (5 pags)...
[!] RAM llena (Pag 4). Ejecutando FIFO...
    > SwapOut: Marco 0 (P1, Pag 0) -> Swap 0
[Paging] Proceso 2 listo.
[Exito] Proceso 2 cargado en memoria.

Eliminando proceso 1...
[Paging] Proceso 1 finalizado y memoria limpia.
[Simulador] Proceso 1 terminado correctamente.
Proceso eliminado correctamente.

Estado de RAM/SWAP:

--- ESTADO DE RAM ---
[Frame 0: P2, Pag 4]
[Frame 1: LIBRE]
[Frame 2: LIBRE]
[Frame 3: LIBRE]
[Frame 4: P2, Pag 0]
[Frame 5: P2, Pag 1]
[Frame 6: P2, Pag 2]
[Frame 7: P2, Pag 3]

--- ESTADO DE SWAP ---
[Frame 0: LIBRE]
[Frame 1: LIBRE]
[Frame 2: LIBRE]
[Frame 3: LIBRE]
[Frame 4: LIBRE]
[Frame 5: LIBRE]
[Frame 6: LIBRE]
[Frame 7: LIBRE]
[Frame 8: LIBRE]
[Frame 9: LIBRE]
[Frame 10: LIBRE]
[Frame 11: LIBRE]
[Frame 12: LIBRE]
[Frame 13: LIBRE]
[Frame 14: LIBRE]
[Frame 15: LIBRE]

✓ TODOS LOS TESTS SE EJECUTARON (revisa consola para ver detalles).
```

Test 5 — test_5_process_termination

Simulador

Flujo básico

```
kim@kims-MacBook-Air src % python3 ui.py
Iniciando Sistema Operativo...

=====
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: |
```

Menú principal

```
Selecciona una opción: 1
Tamaño del proceso (bytes): 200

[Simulador] Creando Proceso 1 (200 bytes)...
[Simulador] Requiere 1 páginas.
[Paging] Cargando proceso 1 (1 pags)...
[Paging] Proceso 1 listo.
[Exito] Proceso 1 cargado en memoria.

[Presiona ENTER para continuar...]
```

Opción 1 — Creación de nuevo proceso

Selecciona una opción: 3

--- ESTADO DE RAM---

[Frame 0: P1, Pag 0]
[Frame 1: LIBRE]
[Frame 2: LIBRE]
[Frame 3: LIBRE]
[Frame 4: LIBRE]
[Frame 5: LIBRE]
[Frame 6: LIBRE]
[Frame 7: LIBRE]

--- ESTADO DE SWAP---

[Frame 0: LIBRE]
[Frame 1: LIBRE]
[Frame 2: LIBRE]
[Frame 3: LIBRE]
[Frame 4: LIBRE]
[Frame 5: LIBRE]
[Frame 6: LIBRE]
[Frame 7: LIBRE]
[Frame 8: LIBRE]
[Frame 9: LIBRE]
[Frame 10: LIBRE]
[Frame 11: LIBRE]
[Frame 12: LIBRE]
[Frame 13: LIBRE]
[Frame 14: LIBRE]
[Frame 15: LIBRE]

[Presiona ENTER para continuar...]

Opción 3—Estado de RAM y SWAP

Selecciona una opción: 4

ID del Proceso: 1

--- Tabla de Paginas: Proceso ID 1 ---

Tamaño: 200 bytes | Total Paginas: 1

Pagina	Ubicacion	Marco Fisico	En RAM
0	RAM	0	Si

[Presiona ENTER para continuar...]

Opción 4 — Tabla de páginas de proceso

Selecciona una opción: 5

--- ESTADÍSTICAS DEL SISTEMA ---

- > Marcos RAM Libres: 7 / 8
- > Uso de Swap: 0 / 16
- > Fallos de Página: 0

[Presiona ENTER para continuar...]

Opción 5 — Estadísticas globales

Selecciona una opción: 2

ID del Proceso a terminar: 1

[Paging] Proceso 1 finalizado y memoria limpiada.

[Simulador] Proceso 1 terminado correctamente.

[Presiona ENTER para continuar...]

Opción 2 — Proceso terminado

Memoria RAM llena

```
Selecciona una opción: 1
Tamaño del proceso (bytes): 900

[Simulador] Creando Proceso 5 (900 bytes)...
[Simulador] Requiere 4 páginas.
[Paging] Cargando proceso 5 (4 pags)...
[!] RAM llena (Pag 2). Ejecutando FIFO...
    > SwapOut: Marco 0 (P2, Pag 0) -> Swap 0
[!] RAM llena (Pag 3). Ejecutando FIFO...
    > SwapOut: Marco 1 (P3, Pag 0) -> Swap 1
[Paging] Proceso 5 listo.
[Exito] Proceso 5 cargado en memoria.

[Presiona ENTER para continuar...]
```

Memoria RAM llena, se ejecuta FIFO

Selecciona una opción: 3

--- ESTADO DE RAM---

- [Frame 0: P5, Pag 2]
- [Frame 1: P5, Pag 3]
- [Frame 2: P3, Pag 1]
- [Frame 3: P3, Pag 2]
- [Frame 4: P4, Pag 0]
- [Frame 5: P4, Pag 1]
- [Frame 6: P5, Pag 0]
- [Frame 7: P5, Pag 1]

--- ESTADO DE SWAP---

- [Frame 0: P2, Pag 0]
- [Frame 1: P3, Pag 0]
- [Frame 2: LIBRE]
- [Frame 3: LIBRE]
- [Frame 4: LIBRE]
- [Frame 5: LIBRE]
- [Frame 6: LIBRE]
- [Frame 7: LIBRE]
- [Frame 8: LIBRE]
- [Frame 9: LIBRE]
- [Frame 10: LIBRE]
- [Frame 11: LIBRE]
- [Frame 12: LIBRE]
- [Frame 13: LIBRE]
- [Frame 14: LIBRE]
- [Frame 15: LIBRE]

[Presiona ENTER para continuar...]

Estado de RAM y SWAP con procesos 2 y 3 en SWAP

Selecciona una opción: 4

ID del Proceso: 5

--- Tabla de Paginas: Proceso ID 5 ---

Tamaño: 900 bytes | Total Páginas: 4

Página	Ubicación	Marco Físico	En RAM
0	RAM	6	Si
1	RAM	7	Si
2	RAM	8	Si
3	RAM	1	Si

[Presiona ENTER para continuar...]

Tabla de páginas del último proceso agregado: Proceso 5

Selecciona una opción: 5

--- ESTADÍSTICAS DEL SISTEMA ---

- > Marcos RAM Libres: 0 / 8
- > Uso de Swap: 2 / 16
- > Fallos de Página: 2

[Presiona ENTER para continuar...]

Estadísticas globales muestran memoria RAM llena, 2 usos de SWAP y 2 fallos de página

Selecciona una opción: 2

ID del Proceso a terminar: 5

[Paging] Proceso 5 finalizado y memoria limpiada.

[Simulador] Proceso 5 terminado correctamente.

[Presiona ENTER para continuar...]

Proceso 5 terminado

Selecciona una opción: 3

--- ESTADO DE RAM---

- [Frame 0: LIBRE]
- [Frame 1: LIBRE]
- [Frame 2: P3, Pag 1]
- [Frame 3: P3, Pag 2]
- [Frame 4: P4, Pag 0]
- [Frame 5: P4, Pag 1]
- [Frame 6: LIBRE]
- [Frame 7: LIBRE]

--- ESTADO DE SWAP---

- [Frame 0: P2, Pag 0]
- [Frame 1: P3, Pag 0]
- [Frame 2: LIBRE]
- [Frame 3: LIBRE]
- [Frame 4: LIBRE]
- [Frame 5: LIBRE]
- [Frame 6: LIBRE]
- [Frame 7: LIBRE]
- [Frame 8: LIBRE]
- [Frame 9: LIBRE]
- [Frame 10: LIBRE]
- [Frame 11: LIBRE]
- [Frame 12: LIBRE]
- [Frame 13: LIBRE]
- [Frame 14: LIBRE]
- [Frame 15: LIBRE]

[Presiona ENTER para continuar...]

Marcos libres de la memoria RAM

Selecciona una opción: 4

ID del Proceso: 3

--- Tabla de Paginas: Proceso ID 3 ---

Tamaño: 600 bytes | Total Paginas: 3

Pagina	Ubicacion	Marco Fisico	En RAM
--------	-----------	--------------	--------

0	SWAP	1	No
1	RAM	2	Si
2	RAM	3	Si

[Presiona ENTER para continuar...]

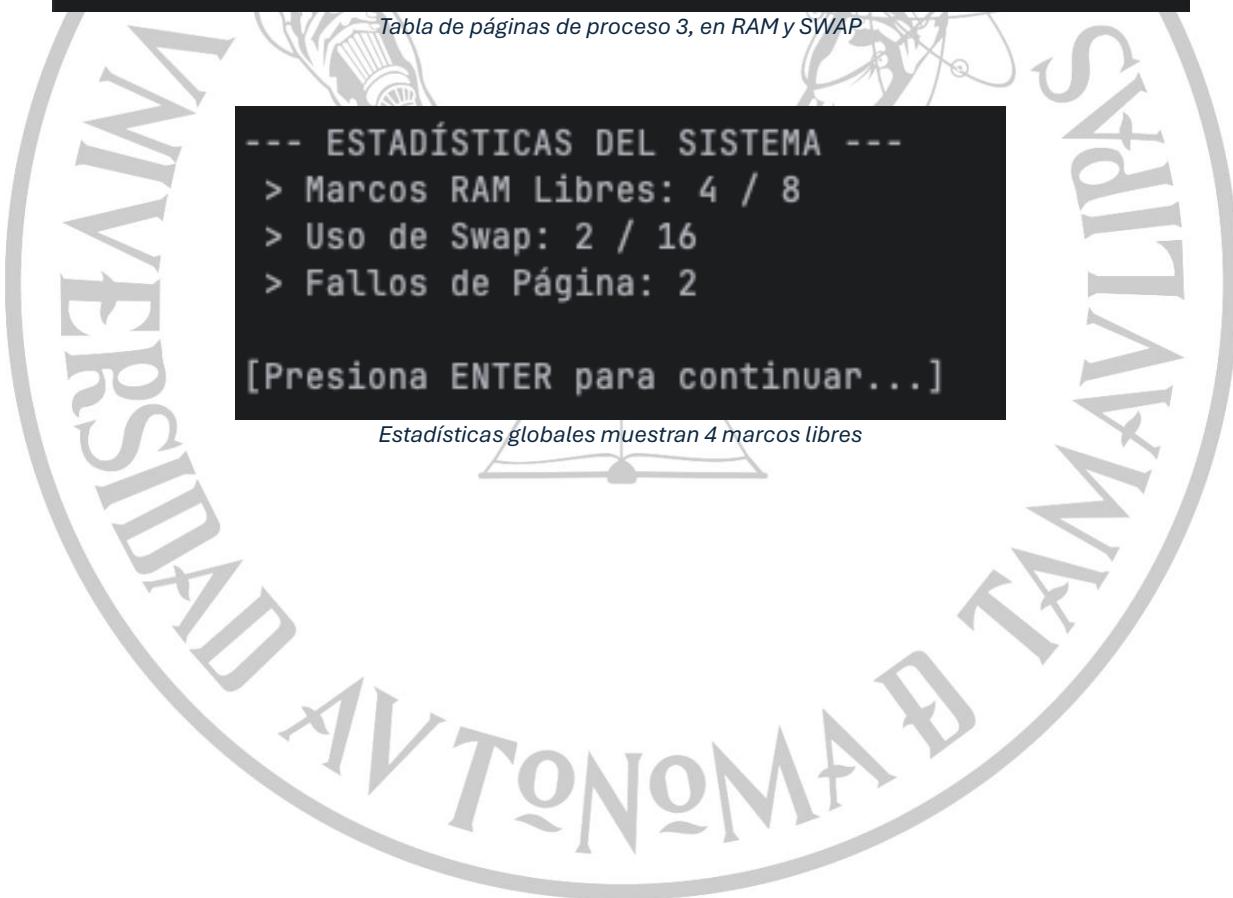
Tabla de páginas de proceso 3, en RAM y SWAP

--- ESTADÍSTICAS DEL SISTEMA ---

- > Marcos RAM Libres: 4 / 8
- > Uso de Swap: 2 / 16
- > Fallos de Página: 2

[Presiona ENTER para continuar...]

Estadísticas globales muestran 4 marcos libres



Captura de errores y excepciones

```
Selecciona una opción: 1
Tamaño del proceso (bytes): 400

[Simulador] Creando Proceso 13 (400 bytes)...
[Simulador] Requiere 2 páginas.
[Paging] Cargando proceso 13 (2 pags)...
  [!] RAM llena (Pag 0). Ejecutando FIFO...
    > SwapOut: Marco 1 (P9, Pag 0) -> Swap 15
  [!] RAM llena (Pag 1). Ejecutando FIFO...
Error: Swap lleno. No se puede realizar intercambio.
[Error] Fallo al cargar Proceso 13 (Memoria llena o error interno).

[Presiona ENTER para continuar...]
```

RAM y SWAP llenas

```
Selecciona una opción: 1
Tamaño del proceso (bytes): -10
El tamaño debe ser mayor a 0.

[Presiona ENTER para continuar...]
```

Intento de creación de proceso con tamaño negativo

```
Selecciona una opción: 2
ID del Proceso a terminar: 20
[Error] El proceso 20 no existe.

[Presiona ENTER para continuar...]
```

Intento de terminación de proceso con PID 20

```
1  [MEMORY]
2  ram_size = 2048asd
3  swap_size = 4096
4  page_size = 256
```

```
kim@kims-MacBook-Air src % python3 ui.py
Iniciando Sistema Operativo...
Error: los valores de [MEMORY] deben ser números enteros.
```

Valores inválidos dentro de config.ini

Código fuente

```
config_loader.py
import configparser
import os
import sys

def load_config(filename = 'config.ini'):
    config = configparser.ConfigParser(inline_comment_prefixes=('#', ';'))

    if not os.path.exists(filename):
        print(f"Error: El archivo {filename} no existe.")
        sys.exit(1)

    files = config.read(filename)
    if len(files) == 0:
        print(f"Error: No se pudo leer el archivo {filename}.")
        sys.exit(1)

    if 'MEMORY' not in config:
        print("Error: el archivo config.ini no contiene la sección [MEMORY].")
        sys.exit(1)

    section = config['MEMORY']
    required_keys = ['ram_size', 'swap_size', 'page_size']

    for key in required_keys:
        if key not in section:
            print(f"Error: no se encontro la clave '{key}' dentro de [MEMORY].")
            sys.exit(1)
```

```
try:  
    ram_size = int(section['ram_size'])  
    swap_size = int(section['swap_size'])  
    page_size = int(section['page_size'])  
except ValueError:  
    print("Error: los valores de [MEMORY] deben ser números enteros.")  
    sys.exit(1)  
  
if ram_size <= 0 or swap_size <= 0 or page_size <= 0:  
    print("Error: todos los valores deben ser mayores a cero.")  
    sys.exit(1)  
  
if ram_size % page_size != 0:  
    print("Error: RAM no es múltiplo exacto...")  
    sys.exit(1)  
if swap_size % page_size != 0:  
    print("Error: RAM no es múltiplo exacto...")  
    sys.exit(1)  
  
frames_ram = ram_size // page_size  
frames_swap = swap_size // page_size  
  
cfg = {  
    "ram_size": ram_size,  
    "swap_size": swap_size,  
    "page_size": page_size,  
    "frames_ram": frames_ram,  
    "frames_swap": frames_swap  
}  
  
return cfg
```

```
if __name__ == '__main__':
    config = load_config()
    print("CONFIG LOADED:", config)

process.py
import math

class Process:
    def __init__(self, pid, size, page_size=256):
        # Mantenemos compatibilidad con todos:
        self.id = pid # Para que simulator.py no llore
        self.pid = pid # Para que paging.py no llore

        self.state = "new"
        self.size = size
        self.page_size = page_size

        # Calcular páginas necesarias
        self.num_pages = math.ceil(size / page_size)

        # Inicializar tabla de páginas
        self.page_table = {}
        self.init_page_table()

    def init_page_table(self):
        for i in range(self.num_pages):
            self.page_table[i] = {
```

```

        'frame_id': None,
        'location': None, # 'RAM', 'SWAP' o None
        'present': False # True solo si esta en RAM
    }

def update_page_status(self, page_number, frame_id, location):
    if page_number in self.page_table:
        self.page_table[page_number]['frame_id'] = frame_id
        self.page_table[page_number]['location'] = location
        self.page_table[page_number]['present'] = (location == 'RAM')
        self.update_state()
    else:
        print(f"Error: La pagina {page_number} no existe en el proceso {self.pid}")

def show_page_table(self):
    print(f"\n--- Tabla de Paginas: Proceso ID {self.pid} ---")
    print(f"Tamaño: {self.size} bytes | Total Paginas: {self.num_pages}")
    print("-" * 60)
    print(f"{'Pagina':<10} | {'Ubicacion':<10} | {'Marco Fisico':<15} | {'En RAM':<10}")
    print("-" * 60)

    for page, info in self.page_table.items():
        loc = info['location'] if info['location'] else "Pendiente"
        fid = info['frame_id'] if info['frame_id'] is not None else "-"
        present = "Si" if info['present'] else "No"

        print(f"{page:<10} | {loc:<10} | {fid:<15} | {present:<10}")
    print("-" * 60)

def __repr__(self):
    return f"<Proceso {self.pid}: {self.size} bytes ({self.num_pages} pags)>"

```

```
def update_state(self):
    in_ram = any(info['location'] == 'RAM' for info in self.page_table.values())
    in_swap = any(info['location'] == 'SWAP' for info in self.page_table.values())
```

```
if in_ram and not in_swap:
    self.state = "active"
elif in_ram and in_swap:
    self.state = "partially_swapped"
elif not in_ram and in_swap:
    self.state = "swapped"
else:
    self.state = "new"
```

paging.py

```
class PagingSystem:
```

```
    def __init__(self, memory_manager, page_size):
```

```
        """
```

Inicializa el sistema de paginación.

:param memory_manager: Instancia de la clase MemoryManager (memory.py)

:param page_size: Tamaño de página definido en config.ini

```
        """
```

```
        self.memory = memory_manager
```

```
        self.page_size = page_size
```

```
        self.active_processes = {}
```

```
        self.page_faults = 0
```

```
    def load_process(self, process):
```

```
        """
```

Carga un proceso en memoria. Si la RAM está llena, aplica FIFO.

```
"""
print(f"[Paging] Cargando proceso {process.pid} ({process.num_pages} pags)...")  
# Registrar proceso en active_processes  
self.active_processes[process.pid] = process  
  
for i in range(process.num_pages):  
    # 1. Intentar asignar en RAM  
    frame_id, error = self.memory.allocate_ram(process.pid, i)  
  
    if error == "NO_SPACE":  
        # Contar esto como un page fault (se requiere swapping)  
        self.page_faults += 1  
  
        print(f"  [!] RAM llena (Pag {i}). Ejecutando FIFO...")  
  
        # A) Seleccionar víctima  
        victim_frame_id = self.memory.select_victim_fifo()  
        if victim_frame_id is None:  
            print("Error: No se pudo seleccionar víctima para Swap.")  
            return False  
  
        # B) Identificar dueño del marco  
        victim_frame = self.memory.ram[victim_frame_id]  
        victim_pid = victim_frame.process_id  
        victim_page_num = victim_frame.page_number  
  
        # C) Mover a Swap  
        swap_id, swap_err = self.memory.move_ram_to_swap(victim_frame_id)  
        if swap_err:  
            print(f"Error: Swap lleno. No se puede realizar intercambio.")  
            return False
```

```

print(
    f"    > SwapOut: Marco {victim_frame_id} (P{victim_pid}, Pag
{victim_page_num}) -> Swap {swap_id}")

# D) Actualizar tabla de páginas de la víctima (si existe)
if victim_pid in self.active_processes:
    self.active_processes[victim_pid].update_page_status(victim_page_num,
swap_id, 'SWAP')

# E) Asignar el marco liberado al proceso actual
frame_id, error = self.memory.allocate_ram(process.pid, i)
if error:
    # si falla de nuevo, abortamos
    print(f"Error: no se pudo asignar marco tras swap-out (proc {process.pid},
pag {i}).")
    return False

# Si llegamos aquí, frame_id es válido y la página queda en RAM
process.update_page_status(i, frame_id, 'RAM')

print(f"[Paging] Proceso {process.pid} listo.")
return True

def unload_process(self, pid):
"""
Termina un proceso y libera TODOS sus recursos (RAM y Swap).
"""

if pid in self.active_processes:
    process = self.active_processes[pid]

```

```

# Recorremos la tabla para liberar lo que tenga asignado
for page_num, info in process.page_table.items():

    # Liberar RAM
    if info['location'] == 'RAM' and info['frame_id'] is not None:
        self.memory.free_ram_frame(info['frame_id'])

    # Liberar SWAP (Corrección del Bloque 3)
    elif info['location'] == 'SWAP' and info['frame_id'] is not None:
        # Accedemos directo a la lista de swap del manager para liberarlo
        # ya que memory.py no tiene un metodo 'free_swap_frame' explícito,
        # usamos el método del objeto Frame.
        if info['frame_id'] < len(self.memory.swap):
            self.memory.swap[info['frame_id']].free_frame()

    # marcar proceso como terminado (por si algún módulo lo consulta antes de
    # borrarlo)
    process.state = "terminated"

    # limpiar la estructura y eliminar del registro
    del self.active_processes[pid]
    print(f"[Paging] Proceso {pid} finalizado y memoria limpia.")

else:
    print(f"Advertencia: El proceso {pid} no existe.")

def swap_in(self, pid, page_number):
    """
    Trae una página desde SWAP a RAM (swap-in).
    Retorna True si tuvo éxito, False si falló.
    """

    if pid not in self.active_processes:

```

Trae una página desde SWAP a RAM (swap-in).

Retorna True si tuvo éxito, False si falló.

"""

```
if pid not in self.active_processes:
```

```
print(f"[swap_in] Proceso {pid} no está registrado.")
return False

proc = self.active_processes[pid]
entry = proc.page_table.get(page_number)
if entry is None:
    print(f"[swap_in] Página {page_number} fuera de rango para P{pid}.")
    return False

# Si ya está en RAM, nada que hacer
if entry['location'] == 'RAM' and entry['frame_id'] is not None:
    return True

# Debe estar en swap (si no, es un caso lógico)
swap_slot = entry.get('frame_id')
if swap_slot is None:
    print(f"[swap_in] La página {page_number} de P{pid} no está en SWAP.")
    return False

# Intentar asignar un marco en RAM
frame_id, err = self.memory.allocate_ram(pid, page_number)
if err == "NO_SPACE":
    # page fault por swap-in: contar y expulsar víctima
    self.page_faults += 1
    victim = self.memory.select_victim_fifo()
    if victim is None:
        print("[swap_in] No se pudo seleccionar víctima para swap-in.")
        return False

    victim_frame = self.memory.ram[victim]
    victim_pid = victim_frame.process_id
```

```
        victim_page = victim_frame.page_number

        # Mover víctima a swap
        new_swap_id, swap_err = self.memory.move_ram_to_swap(victim)
        if swap_err:
            print("[swap_in] Swap lleno al intentar swap-in.")
            return False

        # Actualizar tabla de la víctima
        if victim_pid in self.active_processes:
            self.active_processes[victim_pid].update_page_status(victim_page,
new_swap_id, 'SWAP')

        # Reintentar asignación
        frame_id, err = self.memory.allocate_ram(pid, page_number)
        if err:
            print("[swap_in] Error: no se pudo asignar marco tras swap-out.")
            return False

        # Ahora tenemos frame_id en RAM; liberar la entrada en swap
        # Liberar el marco en swap (si existe)
        if swap_slot < len(self.memory.swap):
            self.memory.swap[swap_slot].free_frame()

        # Actualizar tabla del proceso
        proc.update_page_status(page_number, frame_id, 'RAM')

        # Contar el swap-in como un page fault también (opcional, pero usual)
        self.page_faults += 1
```

```
        print(f"[swap_in] Página {page_number} de P{pid} traída de Swap {swap_slot} a  
RAM marco {frame_id}.")  
    return True
```

```
def print_page_table(self, pid):  
    if pid in self.active_processes:  
        self.active_processes[pid].show_page_table()  
    else:  
        print(f"Proceso {pid} no encontrado.")
```

memory.py

```
from collections import deque
```

```
class Frame:  
    def __init__(self, frame_id):  
        self.id = frame_id # número de marco  
        self.process_id = None # id de proceso  
        self.page_number = None # número de página de proces  
        self.free = True # libre al inicio
```

```
# asignar página al marco
```

```
def assign(self, process_id, page_number):  
    self.process_id = process_id  
    self.page_number = page_number  
    self.free = False
```

```
# liberación de marco
```

```
def free_frame(self):  
    self.process_id = None
```

```
    self.page_number = None  
    self.free = True  
  
def __repr__(self):  
    if self.free:  
        return f"[Frame {self.id}: LIBRE]"  
    return f"[Frame {self.id}: P{self.process_id}, Pag {self.page_number}]"
```

```
class MemoryManager:  
    def __init__(self, frames_ram, frames_swap):  
        self.ram = [Frame(i) for i in range(frames_ram)]  
        self.swap = [Frame(i) for i in range(frames_swap)]  
  
        # FIFO  
        self fifo_queue = deque()  
  
    #-----#  
    # MÉTODOS PARA RAM #  
    #-----#  
  
    def get_free_ram_frame(self):  
        for frame in self.ram:  
            if frame.free:  
                return frame.id  
        return None  
  
    def allocate_ram(self, process_id, page_number):  
        free_id = self.get_free_ram_frame()  
  
        if free_id is not None:
```

```
frame = self.ram[free_id]
frame.assign(process_id, page_number)

self fifo queue.append(free_id)

return free_id, None
else:
    return None, "NO_SPACE"

def free_ram_frame(self, frame_id):
    frame = self.ram[frame_id]
    frame.free_frame()

    if frame_id in self fifo queue:
        self fifo queue.remove(frame_id)

# -----#
# ALGORITMO DE REEMPLAZO FIFO #
# -----#

def select_victim_fifo(self):
    if not self fifo queue:
        return None

    victim_frame_id = self fifo queue.popleft()
    return victim_frame_id

# -----#
# MÉTODOS PARA SWAP #
# -----#
```

```
def get_free_swap_frame(self):
    for frame in self.swap:
        if frame.free:
            return frame.id
    return None

def move_ram_to_swap(self, frame_id):
    swap_id = self.get_free_swap_frame()
    if swap_id is None:
        return None, "SWAP_FULL"

    ram_frame = self.ram[frame_id]
    swap_frame = self.swap[swap_id]

    swap_frame.assign(ram_frame.process_id, ram_frame.page_number)

    ram_frame.free_frame()

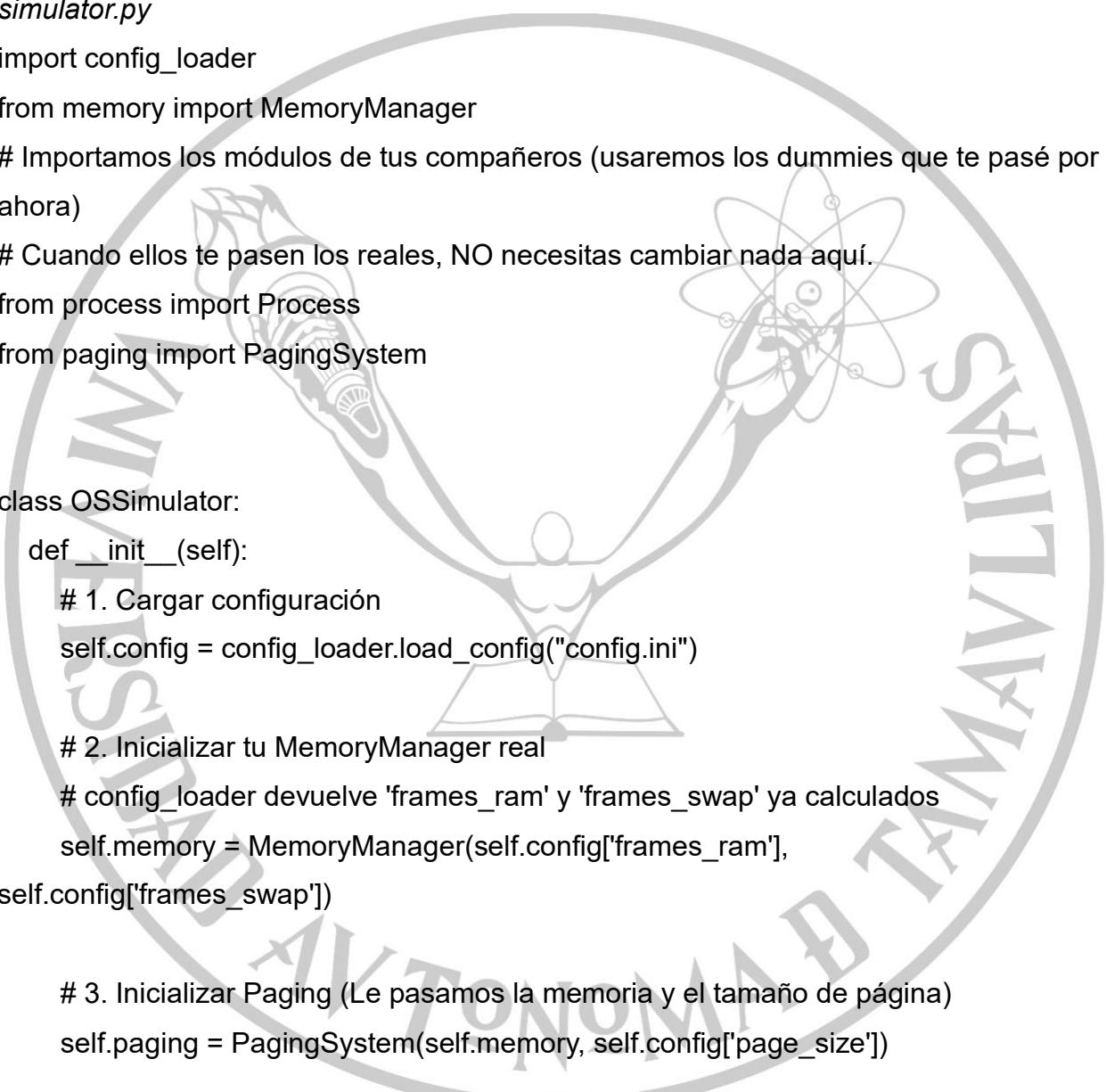
    return swap_id, None

# -----
# VISUALIZACIÓN #
# -----#
```

```
def print_ram(self):
    print("\n--- ESTADO DE RAM---")
    for frame in self.ram:
        print(frame)

def print_swap(self):
    print("\n--- ESTADO DE SWAP---")
```

```
for frame in self.swap:  
    print(frame)
```



```
simulator.py  
import config_loader  
from memory import MemoryManager  
# Importamos los módulos de tus compañeros (usaremos los dummies que te pasé por  
ahora)  
# Cuando ellos te pasen los reales, NO necesitas cambiar nada aquí.  
from process import Process  
from paging import PagingSystem  
  
class OSSimulator:  
    def __init__(self):  
        # 1. Cargar configuración  
        self.config = config_loader.load_config("config.ini")  
  
        # 2. Inicializar tu MemoryManager real  
        # config_loader devuelve 'frames_ram' y 'frames_swap' ya calculados  
        self.memory = MemoryManager(self.config['frames_ram'],  
                                    self.config['frames_swap'])  
  
        # 3. Inicializar Paging (Le pasamos la memoria y el tamaño de página)  
        self.paging = PagingSystem(self.memory, self.config['page_size'])  
  
        self.processes = {} # Diccionario para guardar procesos activos {pid: Process}  
        self.pid_counter = 1  
  
    def create_process(self, size_bytes):
```

```
# Crear proceso (Lógica de Roberto)
new_process = Process(self.pid_counter, size_bytes)

print(f"\n[Simulador] Creando Proceso {new_process.id} ({size_bytes} bytes)...")
print(f"[Simulador] Requiere {new_process.num_pages} páginas.")

# Cargar proceso usando Paging (Lógica de Sergio)
success = self.paging.load_process(new_process)

if success:
    self.processes[self.pid_counter] = new_process
    print(f"[Exito] Proceso {self.pid_counter} cargado en memoria.")
    self.pid_counter += 1
else:
    print(f"[Error] Fallo al cargar Proceso {self.pid_counter} (Memoria llena o error interno.)")

def terminate_process(self, pid):
    if pid in self.processes:
        # Paging se encarga de liberar marcos
        self.paging.unload_process(pid)
        del self.processes[pid]
        print(f"[Simulador] Proceso {pid} terminado correctamente.")
    else:
        print(f"[Error] El proceso {pid} no existe.")

def show_memory_status(self):
    # Usamos tus métodos de visualización de memory.py
    self.memory.print_ram()
    self.memory.print_swap()
```

```
def show_page_table(self, pid):
    if pid in self.processes:
        # Esto lo maneja Sergio en Paging
        self.paging.print_page_table(pid)
    else:
        print("Proceso no encontrado.")

def get_global_stats(self):
    # 1. Calcular Marcos Libres en RAM
    # Accedemos a la lista .ram de tu MemoryManager y contamos los .free
    free_ram = sum(1 for frame in self.memory.ram if frame.free)
    total_ram = len(self.memory.ram)

    # 2. Calcular Uso de Swap
    # Contamos cuántos marcos de swap están ocupados (no free)
    used_swap = sum(1 for frame in self.memory.swap if not frame.free)
    total_swap = len(self.memory.swap)

    # 3. Fallos de página (Page Faults)
    # Intentamos leerlo del sistema de paginación (si Sergio lo implementa)
    # Si no existe (porque usamos el dummy), mostramos 0
    page_faults = getattr(self.paging, 'page_faults', 0)

    return {
        "Marcos RAM Libres": f"{free_ram} / {total_ram}",
        "Uso de Swap": f"{used_swap} / {total_swap}",
        "Fallos de Página": page_faults
    }
```

ui.py

```
from simulator import OSSimulator  
import os  
import sys
```

```
def clear_console():  
    # Limpia pantalla según windows o linux  
    os.system('cls' if os.name == 'nt' else 'clear')  
  
def main():  
    print("Iniciando Sistema Operativo...")  
    # Aseguramos que cargue la configuración correctamente al inicio  
    try:  
        sim = OSSimulator()  
    except Exception as e:  
        print(f"Error fatal al iniciar simulador: {e}")  
        return  
  
    while True:  
        print("\n" + "=" * 40)  
        print("    SIMULADOR DE MEMORIA (FIFO)")  
        print("=" * 40)  
        print("1. Crear nuevo proceso")  
        print("2. Terminar proceso")  
        print("3. Ver estado de RAM y SWAP")  
        print("4. Ver tabla de páginas de un proceso")  
        print("5. Ver estadísticas globales")  
        print("6. Salir")  
        print("-" * 40)
```

```
opcion = input("Selecciona una opción: ")

if opcion == '1':
    try:
        size = int(input("Tamaño del proceso (bytes): "))
        if size > 0:
            sim.create_process(size)
        else:
            print("El tamaño debe ser mayor a 0.")
    except ValueError:
        print("Por favor, ingresa un número válido.")

elif opcion == '2':
    try:
        pid = int(input("ID del Proceso a terminar: "))
        sim.terminate_process(pid)
    except ValueError:
        print("ID inválido (debe ser numérico).")

elif opcion == '3':
    sim.show_memory_status()

elif opcion == '4':
    try:
        pid = int(input("ID del Proceso: "))
        sim.show_page_table(pid)
    except ValueError:
        print("ID inválido.")

elif opcion == '5':
```

```
stats = sim.get_global_stats()
print("\n--- ESTADÍSTICAS DEL SISTEMA ---")
for key, value in stats.items():
    print(f" > {key}: {value}")

elif opcion == '6':
    print("Apagando simulador...")
    break
else:
    print("Opción no reconocida.")

input("\n[Presiona ENTER para continuar...]")
clear_console()

if __name__ == "__main__":
    main()

test.py
import config_loader
from simulator import OSSimulator

def separator(title):
    print("\n" + "=" * 60)
    print(f"TEST: {title}")
    print("=" * 60)

def test_1_basic_process_load():
```

```
separator("1) Carga básica de procesos")
```

```
sim = OSSimulator()
```

```
sim.create_process(200)
```

```
sim.create_process(300)
```

```
stats = sim.get_global_stats()
```

```
print("\nEstadísticas:", stats)
```

```
assert isinstance(stats["Fallos de Página"], int)
```

```
def test_2_force_swapping():
```

```
separator("2) Forzar swapping llenando RAM")
```

```
sim = OSSimulator()
```

```
# RAM = 8 frames → este proceso fuerza swapping
```

```
sim.create_process(3000) # ≈ 12 páginas
```

```
print("\nVer estado final de RAM:")
```

```
sim.show_memory_status()
```

```
stats = sim.get_global_stats()
```

```
print("\nEstadísticas:", stats)
```

```
assert sim.paging.page_faults > 0, "Debe haber fallos de página al activar FIFO"
```

```
def test_3_page_table_integrity():
```

```
separator("3) Integridad de tabla de páginas")
```

```
sim = OSSimulator()

psize = 700
sim.create_process(psize)

pid = 1
print("\nTabla de páginas del proceso:")
sim.show_page_table(pid)

proc = sim.processes[pid]

assert len(proc.page_table) == proc.num_pages
assert proc.state in ["active", "partially_swapped", "swapped"]

def test_4_swap_in_operation():
    separator("4) Test explícito de swap-in")

    sim = OSSimulator()

    sim.create_process(1500) # 6 páginas
    pid = 1
    proc = sim.processes[pid]

    swap_pages = [
        p for p, info in proc.page_table.items()
        if info["location"] == "SWAP"
    ]

    if not swap_pages:
```

```
print("No hubo páginas en swap. Cambiar tamaños para test.")
return

page_toBring = swap_pages[0]
print(f"\nIntentando swap-in de página {page_toBring} del proceso {pid}...")
result = sim.paging.swap_in(pid, page_toBring)

assert result is True, "swap_in debe funcionar"
assert proc.page_table[page_toBring]["location"] == "RAM"
print("swap_in exitoso.")

def test_5_process_termination():
    separator("5) Terminación de proceso y limpieza")

    sim = OSSimulator()

    sim.create_process(900) # varias páginas
    sim.create_process(1200)

    print("\nEliminando proceso 1...")
    sim.terminate_process(1)

    assert 1 not in sim.processes
    print("Proceso eliminado correctamente.")

    print("\nEstado de RAM/SWAP:")
    sim.show_memory_status()

# -----
```

```
# EJECUCIÓN DIRECTA DE TESTS
# -----
if __name__ == "__main__":
    separator("INICIANDO PRUEBAS DEL SISTEMA OPERATIVO")

    test_1_basic_process_load()
    test_2_force_swapping()
    test_3_page_table_integrity()
    test_4_swap_in_operation()
    test_5_process_termination()

    print("\n✓ TODOS LOS TESTS SE EJECUTARON (revisa consola para ver
detalles.)"

config.ini
[MEMORY]
ram_size = 2048
swap_size = 4096
page_size = 256
```