OPEN ENDED LAB

Course: CS-329 Operating Systems
Course Instructor: Dr. Zareen Sadiq
Project Title: Dynamic Partitioning Scheme for Memory Management

Group Members:

Sumaika Imran	**CS-22009**
Malaika Mustafa	**CS-22006**
Qurratulain	**CS-22018**

Memory Management Module

1. Introduction

This project focuses on designing and implementing a Memory Management Module for an Operating System, simulating the allocation and de-allocation of memory to processes based on Dynamic Partitioning. The system accepts a list of processes with their arrival time, execution time, and memory size, then allocates memory dynamically as processes arrive. It manages memory fragmentation by maintaining free memory holes and merging adjacent free spaces.

2. Why We Use Dynamic Partitioning

In memory management, there are three major techniques:

- **Fixed Partitioning:** Divides memory into fixed-size blocks. It leads to **internal fragmentation** because a process might not fully use the allocated block.
- **Paging:** Divides memory and processes into small equal-sized units (pages and frames). It eliminates external fragmentation but introduces **overhead** in managing page tables and **does not allocate exact needed memory**.
- **Dynamic Partitioning (Chosen Technique):** Allocates memory **exactly** based on the process size. It reduces internal fragmentation, utilizes memory more efficiently by splitting and merging free spaces ("holes") as processes terminate and free up memory.

3. System Design

- **Total Main Memory Size:** 1024 units (MBs assumed)
- **Memory Management Technique Used:** Dynamic Partitioning
- **Process Details:**
    - **Arrival Time:** Time when the process enters the system.
    - **Execution Time:** Duration the process needs to complete.

- **Size:** Memory requirement (in MBs) for each process.

- **Input Mode:**
  - Inputs are **read from an external input text file** (input.txt) during execution.
  - First line of input text file describes the number of processes and second line onwards, each line contains the details of a process in order of arrival time, execution time and size separated by space.
- **Process Constraints:**
  - Minimum number of processes: 10
  - Each process has randomized arrival times, execution times, and sizes provided through the input text file.

- **Memory State Display:** At every time unit, the program prints:
  - Total Memory
  - Used Memory
  - Free Memory
  - List of Allocated Processes
  - List of Free Holes (Unallocated memory blocks, if any)
  - Suspended (Waiting) Processes (if any)
- **Process Allocation and Execution:**
  - When a process arrives, it tries to allocate to an existing hole if any, on the basis of **first fit algorithm.** In case of no holes present currently, the process is allocated in free memory.
  - If memory is insufficient, the process is suspended (queued).
  - On process completion, its memory is freed, and adjacent holes are merged to form larger free spaces (**Defragmentation).**
  - Suspended processes are retried for memory allocation whenever sufficient space is available.

4. Analysis

- **Number of Processes:** Minimum 10 (user can provide more in the input file)
- **Process**
  - Process Size: dynamically based on the user-provided process sizes.

In general, the system shows efficient memory utilization for varied sizes of processes. However, due to dynamic allocation, minor external fragmentation may still occur, requiring careful hole management.

5. Conclusion

This project successfully demonstrates an efficient Dynamic Partitioning Scheme of Memory Management System. Through dynamic allocation, suspension handling, and effective hole merging, the system ensures optimal utilization of available memory. It adapts to varying process requirements in real time, minimizes fragmentation, and maintains continuous monitoring of memory state. Overall, the simulation highlights key principles of memory management in operating systems, offering a practical and robust solution for dynamic memory handling.