Week 10 - LAQ's

Instructions

Illustrate the implementation of Contiguous memory allocation with neat diagrams

Contiguous Memory Allocation

Contiguous memory allocation is a memory management scheme where each process is allocated a single contiguous block of memory. This method is simple and efficient for small systems, but it can lead to fragmentation issues as processes are loaded and removed from memory.

Key Concepts

1. Memory Partitioning: The main memory is divided into fixed or variable-sized partitions.

2. Process Allocation: Each process is allocated a contiguous block of memory from one of the partitions.

3. Fragmentation: Over time, as processes are loaded and removed, free memory can become fragmented, leading to inefficient use of memory.

Implementation Steps

1. Memory Partitioning: The memory is divided into partitions. These can be fixed-size or variable-size.

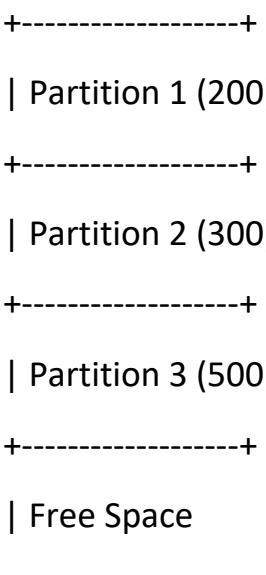
2. Process Loading: When a process is loaded, it is assigned to the first available partition that is large enough to accommodate it.

3. Memory Management: The operating system keeps track of which parts of memory are allocated and which are free.

Diagram of Contiguous Memory Allocation

Initial Memory State

Let's assume we have a memory of size 1000 KB divided into fixed-size partitions:



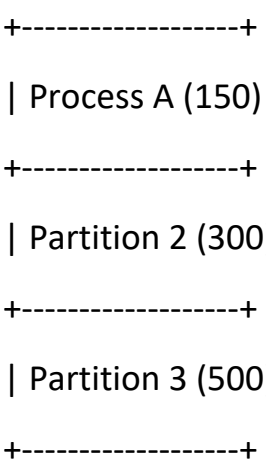
Process Allocation

Now, let's say we have three processes to allocate:

- Process A requires 150 KB
- Process B requires 250 KB
- Process C requires 400 KB

Step 1: Allocate Process A (150 KB)

Process A fits into Partition 1 (200 KB). After allocation, the memory looks like this:



| Free Space (50) |

+------------------+

Step 2: Allocate Process B (250 KB)

Process B fits into Partition 2 (300 KB). After allocation, the memory looks like this:

+------------------+

| Process A (150) |

+------------------+

| Process B (250) |

+------------------+

| Partition 3 (500) |

+------------------+

| Free Space (50) |

+------------------+

Step 3: Allocate Process C (400 KB)

Process C cannot fit into Partition 3 (500 KB) because it is not contiguous with the free space left after Process A and Process B. Therefore, it cannot be allocated.

Fragmentation Example

Now, if Process A terminates, the memory will look like this:

+------------------+

| Free Space (150) |

+------------------+

| Process B (250) |

+------------------+

| Partition 3 (500) |

+------------------+

| Free Space (50) |

+------------------+

In this case, we have internal fragmentation in Partition 1 (50 KB unused) and external fragmentation because the free space is not contiguous enough to accommodate Process C (400 KB).

Summary

Contiguous memory allocation is straightforward and efficient for small systems but can lead to fragmentation issues over time. The operating system must manage memory carefully to minimize fragmentation and ensure efficient allocation of memory to processes.

Advantages and Disadvantages

Advantages:

- Simple to implement and manage.
- Fast access to memory since processes are stored contiguously.

Disadvantages:

- External fragmentation can occur, leading to inefficient memory usage.
- Difficult to allocate memory for larger processes if there is not enough contiguous space, even if total free memory is sufficient.