UNIT VI : MEMORY MANAGEMENT

Memory Management is a critical function of an operating system that handles allocation and deallocation of memory to processes to ensure efficient and secure use of the system's memory resources

Memory Management Requirement :

1. Reallocation : Processes may move in memory during execution; memory management handles relocations dynamically.
2. Protection : Prevent processes from accessing each others memory illegally.
3. Sharing : Allows multiple processes to access the same memory segment for communication or efficiency.
4. Logical Organization : Manage memory into logical modules
5. Physical Organization : Handle mapping between logical memory and physical memory.

MEMORY PARTITIONING

Partitioning divides memory into blocks to allocate it to processes

1. Fixed Partitioning :
   - Memory is divided into fixed-sized partitioning.
   - It is simple to implement
   - But it leads to internal fragmentation.
2. Dynamic Partitioning :
   - Memory is divided dynamically based on process size
   - Avoids internal fragmentation.
   - Causes external fragmentation.
3. Buddy System :
   - Memory is allocated in blocks of size that are power of 2
   - If the required block is not available, larger blocks are split into smaller ones
   - Efficient merging of adjacent free blocks

Fragmentation:

- Internal fragmentation: wasted space within allocated memory blocks.
- External fragmentation: Wasted space in free memory that cannot be used due to fragmentation.

PAGING & SEGMENTATION

1. Paging:
   - divides memory into fixed-sized pages and physical memory into frames
   - Pages are mapped to frames via a page table
   - Eliminates external fragmentation.
   - Introduces page table overhead.
2. Segmentation:
   - divides memory into variable-sized logical segments
   - Matches logical problem structure
   - Causes external fragmentation.
3. Address translation:
   - Virtual Address: generated by CPU
   - Physical Address: actual location in memory.
   - Translation is handled using page table (for paging) and segment table (for segmentation)

PLACEMENT STRATEGIES:

1. First Fit: Allocate first block of memory large enough for process.
2. Best Fit: Allocate the smallest block of memory that is large enough for process
3. Next Fit: similar to first fit but starts search from last allocated block.
4. Worst Fit: Allocate the largest block of memory available to the process.

VIRTUAL MEMORY

Virtual memory allows process to use more ~~mo~~ memory than physically available by storing parts of it on disk

Concepts:

Swapping: Processes are swapped between disk and memory to free up space.

Demand Paging: Loads pages into memory only when they are needed, reducing memory usage.

VIRTUAL MEMORY with PAGING:

- Pages are loaded into memory frames as needed.
- Maintains a page table to map virtual pages to physical frames

Page table structure:

1. Single level page table: simple but uses significant memory. for large processes.
2. Multi level page table: Reduces memory overhead by splitting the page table into smaller tables.
3. Inverted page table: Stores a single entry for each physical frame, reducing memory usage.

TRANSLATION LOOKASIDE BUFFER (TLB)

- A hardware cache for page table entries to speed up address translation.

Page size

larger page size reduces page table overhead but increases internal fragmentation.

VIRTUAL MEMORY with SEGMENTATION.

- combines segmentation with virtual memory to manage logical memory.

combining paging and segmentation:

- Memory is divided into segments and each segment is divided ~~by~~ into pages
- Allows flexible and efficient memory management.

PAGE REPLACEMENT POLICIES

When a page-fault occurs, the system must decide which page to replace.

1. FIFO (First in First out):
- replaces oldest page in memory
- simple to implement
- number of faults is high
- may replace frequently used pages ~~(Belady~~ (Belady anomaly)

2. LRU (Least recently used):
- replaces the page that hasn't been used for the longest time.
- reduces page fault
- Costly to implement.

3. Optimal:
- replaces the page that will not be needed for longest time in future
- Minimum page faults
- Not feasible as future knowledge required.

THRASHING:
- occurs when the system spends most of its time swapping pages instead of executing processes.
- High page fault rates due to insufficient memory allocation.
(solution)
- Reduce degree of multiprogramming or allocate more memory to processes.