**Memory management** -: The task of subdividing the memory among different processes is called memory management. Memory management is a method in the operating system to manage operations between main memory and disk during process execution. The main aim of memory management is to achieve efficient utilisation of memory.

*memory management is done by two method -:*

1. Contiguous memory allocation
2. Single contiguous memory
3. Multiple Partitioning

I. fixed partitioning

ii. dynamic partitioning

1. Non -contiguous memory allocation
2. Paging
3. Segmentation

*Single contiguous memory* -: The main memory is divided into two contiguous areas or partitions. The operating systems reside permanently in one partition, generally at the lower memory, and the user process is loaded into the other partition.

*Multiple Partitioning-:* The operating system needs to divide the available main memory into multiple parts to load multiple processes into the main memory. Thus multiple processes can reside in the main memory simultaneously.

*Fixed Partitioning*-: The main memory is divided into several fixed-sized partitions in a fixed partition memory management scheme or static partitioning. These partitions can be of the same size or different sizes. Each partition can hold a single process.

*Dynamic partitioning* -:In a dynamic partitioning scheme, each process occupies only as much memory as they require when loaded for processing ,and the partitions used are of variable size, and the number of partitions is not defined at the system generation time.

*Paging* -: Paging is a technique that eliminates the requirements of contiguous allocation of main memory. In this, the main memory is divided into fixed-size blocks of physical memory called frames. The size of a frame should be kept the same as that of a page to maximise the main memory and avoid external fragmentation.]

*Segmentation* -: Segmentation is a technique that eliminates the requirements of contiguous allocation of main memory. In this, the main memory is divided into variable-size blocks of physical memory called segments . With segmented memory allocation, each job is divided into several segments of different sizes, one for each module.

**Logical & Physical Address space -:**

*Logical Address space* -:An address generated by the CPU is known as a “Logical Address”. It is also known as a Virtual address. Logical address space can be defined as the size of the process. A logical address can be changed.

*Physical Address space* -:An address seen by the memory unit ,The set of all physical addresses corresponding to these logical addresses is known as Physical address space. A physical address is computed by MMU.

Memory Hierarchy -: Memory Hierarchy is an enhancement to organize the memory such that it can minimise the access time.

Hierarchy Design is divided into 2 main types:

1. *External Memory or Secondary Memory* **–** Comprising Magnetic Disk, Optical Disk, Magnetic Tape i.e. peripheral storage devices which are accessible by the processor via I/O Module.
2. *Internal Memory or Primary Memory –* Comprising Main Memory, Cache Memory & CPU registers. This is directly accessible by the processor.

#### **There are typically four levels of memory in a memory hierarchy:**

1. **Registers**: Registers are small, high-speed memory units located in the CPU. They are used to store the most frequently used data and instructions. Registers have the fastest access time and the smallest storage capacity, typically ranging from 16 to 64 bits.
2. **Cache Memory**: Cache memory is a small, fast memory unit located close to the CPU. It stores frequently used data and instructions that have been recently accessed from the main memory. Cache memory is designed to minimise the time it takes to access data by providing the CPU with quick access to frequently used data.
3. **Main Memory**: Main memory, also known as RAM (Random Access Memory), is the primary memory of a computer system. It has a larger storage capacity than cache memory, but it is slower.
4. **Secondary Storage**: Secondary storage, such as hard disk drives (HDD) and solid-state drives (SSD), is a non-volatile memory unit that has a larger storage capacity than main memory. It is used to store data and instructions that are not currently in use by the CPU.

We can differentiate Memory hierarchy on the basis of this 4 characteristics-:

1. Capacity 2. Access Time

3. Performance 4. Cost Per bit

**Virtual Memory** -: Virtual Memory is a storage allocation scheme in which secondary memory can be addressed as though it were part of the main memory.

Virtual memory is used to run large files then of secondary size. It's also used at a time of multiprogramming.

Demand paging -: The process of loading the page into memory on demand (whenever page fault occurs) is known as demand paging.

Swapping -: swapping is basically a changes of location of pages according to demand from physical memory to secondary memory.

Thrashing -: At a time when multiprogramming is increasing so swapping will also increase.

**Memory allocation & Deallocation** -:

Memory allocation refers to the process of assigning a block of memory to a process or program that requests it. When a program requests memory, the OS checks the availability of free memory and assigns a contiguous block of memory of the requested size to the program.

memory deallocation, also known as memory freeing or deallocation, refers to the process of releasing a block of memory that is no longer needed by a program. When a program releases memory, the OS marks the block of memory as free and adds it to the list of available memory blocks.

**Inter Process Communication (IPC)** -: Interprocess communication (IPC) is a method for two or more processes to communicate and share resources with each other. There are several mechanisms available for IPC, including pipes, message queues, and shared memory.

1. *Pipes:* Pipes are one of the oldest forms of IPC and are used for one-way communication between processes. A pipe is a temporary communication channel that allows data to be passed from one process to another.
2. *Message queues:* Message queues are used for one-way communication between processes and allow messages to be passed between processes without the need for a shared memory area.
3. *Shared memory:* Shared memory is used for two-way communication between processes and allows multiple processes to access a shared region of memory. This region of memory can be used to store data that needs to be accessed by multiple processes.

**Synchronisation techniques**-: These techniques ensure that only one thread or process accesses a shared resource at a time, preventing concurrent access that can cause errors, data corruption, or other unwanted behaviour. Three commonly used synchronisation techniques are mutexes, semaphores, and monitors.

1. *Mutex:* A mutex (short for "mutual exclusion") is a synchronisation object that provides exclusive access to a shared resource. A mutex allows only one thread or process to acquire a lock on the resource at a time. When a thread or process acquires the lock on a mutex, all other threads or processes that attempt to acquire the lock are blocked until the lock is released.
2. *Semaphore:* A semaphore is a synchronisation object that can be used to limit the number of threads or processes that can access a shared resource simultaneously. A semaphore maintains a counter that represents the number of resources available. When a thread or process requests access to the shared resource, the semaphore decrements the counter. If the counter is zero, the semaphore blocks the thread or process until a resource becomes available. When a thread or process releases a resource, the semaphore increments the counter, allowing another thread or process to acquire the resource.
3. *Monitor* : A monitor is a synchronisation construct that allows threads to safely access shared resources by enforcing mutual exclusion and ensuring that only one thread at a time is executing within the monitor. Monitors also provide a mechanism for threads to wait for a particular condition to become true before continuing execution. Monitors are typically implemented using a combination of mutexes, condition variables, and a shared data structure.

->Here's an example program in C++ that demonstrates dynamic memory allocation and deallocation using the new and delete operators.

| #include<bits/stdc++.h>  using namespace std;  int main()  {  int n;  cout<<"Enter the size of array"<<endl;  cin >> n;  int\* arr=new int[n]; //Dynamic allocation of a memory  if(n==0){  cout<<"Memory allocation is failed!"<<endl;  return 0;  }  cout<<"enter a value in array "<<endl;  for(int i=0;i<n;i++)  {  cin>>arr[i];    }  cout<<"array element is -> "<<endl;  for(int i=0;i<n;i++)  {  cout<<"index "<<i<<" = "<<arr[i]<<endl;  }  delete[] arr; //memory cleared  return 0;  } |
| --- |