Solutions to Questions

1. Differentiate between multiprogramming and multiprocessing. 5 marks

Multi programmed System:

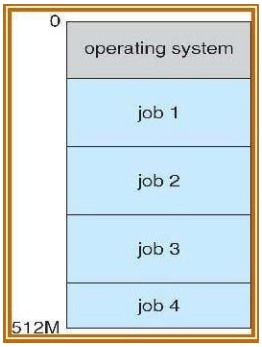
a. If there are two or more programs in the memory at the same time sharing the processor, this is referred as multi programmed OS.

b. It increases the CPU utilization by organizing the jobs so that the CPU will always have one job to execute.

c. Jobs entering the systems are kept in memory.

d. While executing a particular job, if the job has to wait for any task like I/O operation to be complete then the CPU will switch to some other jobs and starts executing it and when the first job finishes waiting the CPU will switch back to that.

e. This will keep the CPU & I/O utilization busy. The following figure shows the memory layout of multiprogrammed OS.



Multiprocessing System:

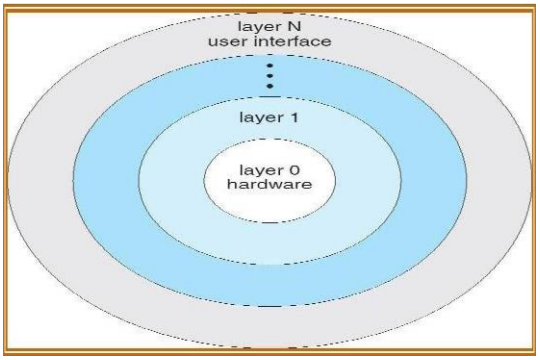
- It is the ability of an OS to execute more than one process simultaneously on a multiprocessor machine. In this, a computer uses more than one CPU at a time.
- Multi processor systems include more than one processor in close communication.
- Two processes can run in parallel.
- Advantages : Increased Throughput, Economy Of Scale, Increased Reliability

PICTURE

2. Short note on Job Pool. 2 M

In general main memory is too small to accommodate all jobs that enters the system. So jobs are kept initially on the disk in the job pool. Job Pool is the one which consists of all processes residing on disk awaiting allocation of main memory. OS selects a job from the job pool, it loads that job into main memory for execution.

3. Explain the advantages of the layered approach. 6M



- In this OS is divided into number of layers, where one layer is built on the top of another layer. The bottom layer is hardware and higher layer is the user interface.
- An OS is an implementation of abstract object i.e. the encapsulation of data & operation to manipulate these data.
- The main advantage of layered approach is the modularity i.e. each layer uses the services & functions provided by the lower layer.
- This approach simplifies the debugging & verification.
- Once first layer is debugged the correct functionality is guaranteed while debugging the second layer. If an error is identified then it is a problem in that layer because the layer below it is already debugged.
- Each layer is designed with only the operations provided by the lower level layers.
- Each layer tries to hide some data structures, operations & hardware from the higher level layers.

4. Difference between a trap and an interrupt. 2M

A **trap**(or an exception) is a software-generated interrupt caused either by an error or by a specific request from a user program that an operating-system service be performed.

An **interrupt** is the occurrence of an event is signalled by an interrupt from either the hardware or the software. Hardware may trigger an interrupt at any time by sending to the CPU, by way of the system bus.

5. What are the essential properties of batch, real time, and distributed OS?

Real time OS:

- Real time system is one that must react to i/p & responds to them quickly.
- A real time system should not be late in response to one event.
- A real time should have well defined time constraints.
- Real time OS uses priority scheduling algorithm to meet the response requirement of a real time application.

Batch OS:

- The user do not interact directly with computers but we use to prepare a job with the program, data & some control information & submit it to the computer operator.
- The job was mainly in the form punched cards.
- At later time the O/P appeared and it consisted of result along with dump of memory and register content for debugging.
- The processing of job was very slow. To improve the processing speed operators batched together the jobs with similar needs and processed it through the computers. This is called Batch Systems.

Distributed OS:

- A distributed system is one in which H/w or S/w components located at the networked computers communicate & co ordinate their actions only by passing messages.
- A distributed systems looks to its user like an ordinary OS but runs on multiple, Independent CPU's.
- Distributed systems depends on networking for their functionality which allows for communication so that distributed systems are able to share computational tasks and provides rich set of features to users.
- N/w may vary by the protocols used, distance between nodes & transport media. Protocols->TCP/IP, ATM etc. Network-> LAN, MAN, WAN etc. Transport Media-> copper wires, optical fibers & wireless transmissions.

6. Is separation of mechanism and policy desirable while designing an OS? Discuss with an example. 8M

   Yes. The separation of mechanism and policy is desirable while designing an OS for flexibility.

   - A general mechanism insensitive to changes in policy would be more desirable.
   - A change in policy would then require redefinition of only certain parameters of the system.

     Example: the timer construct is a mechanism for ensuring CPU protection, but deciding how long the timer is to be set for a particular user is apolicy decision.
   - Microkernel-based OS take the separation of mechanism and policy by implementing a basic set of primitive building blocks.

     As an example, consider the history of UNIX. At first, it had a time-sharing scheduler. In the latest version of Solaries, scheduling is controlled by loadable tables. Depending on the table currently loaded, the system can be time shared, batch processing, real time, fair share or any combination. Making the scheduling mechanism general purpose allows vast policy changes to be made with a single load-new-table command.
   - Policy decisions are important for all resource allocation. Whenever it is necessary to decide whether or not to allocate a resource, a policy decision must be made.

7. What are the different ways in which Pthreads terminates?

   Terminating the thread before it has completed its task is called thread cancellation. The thread to be cancelled is called **target thread**.

   Threads that are no longer needed may be cancelled in one of two ways:

   1. **Asynchronous Cancellation** – One thread immediately terminates the target thread.

      Canceling a thread asynchronously may not free a necessary system-wide resource.
   2. **Deferred Cancellation** – the target thread periodically check whether it has to terminate, thus gives an opportunity to the thread, to terminate itself in an orderly fashion.

      In this method, the operating system will reclaim all the resources before cancellation. With this cancellation, one thread indicates that a target thread is to be cancelled, but cancellation occurs only after the target thread has checked a flag to determine if it should be cancelled or not. This allows a thread to check whether it should be cancelled at a point when it can be cancelled safely. Pthreads refers to such points as **cancellation points.**

8. Explain any two facilities provided for implementing interacting process in programming language and OS.

Command Interpreter

Command Interpreters are used to give commands to the OS. There are multiple command interpreters known as shells. In UNIX and Linux systems, there are several different shells, like the *Bourne shell, C shell, Bourne-Again shell, Korn shell,* and others.

The main function of the command interpreter is to get and execute the user-specified command. Many of the commands manipulate files: create, delete, list, print, copy, execute, and so on.

The commands can be implemented in two general ways-

1) The command interpreter itself contains the code to execute the command.

2) The code to implement the command is in a function in a separate file. The interpreter searches for the file and loads it into the memory and executes it by passing the parameter.

Graphical User Interface, GUI

Another way of interfacing with the operating system is through a user friendly graphical user interface, or GUI. Here, rather than entering commands directly via a command-line interface, users employ a mouse-based window and menu system. The user moves the mouse to position its pointer on images, or icons on the screen (the desktop) that represent

programs, files, directories, and system functions. Depending on the mouse pointer's location, clicking a button on the mouse can invoke a program, select a file or directory-known as a folder-or pull down a menu that contains commands.

1. Differentiate between Direct and Indirect inter process communication. 4M

Direct Communication : In the Direct Communication, each process that wants to communicate must explicitly name the recipient or sender of the communication. In this scheme, the **send and receive** primitives are defined as follows :

Send (P, message) - Send a message to process P.

Receive (Q, message) - Receive a message from process Q.

Indirect Communication : With Indirect Communication, the messages are sent to and received from mailboxes. A mailbox can be viewed abstractly as, an object into which messages can be placed by processes and from which messages can be removed. The send and receive primitives are defined as follows :

Send (A, message) - Send a message to mailbox A.

Received (A, message) - Receive a message from mailbox A.

2. Differentiate between:
3.

I. Process and Thread

A ***process*** is a program in execution.

A ***thread*** is a basic unit of CPU utilization.

Unlike processes, threads are not independent of one another.

Unlike processes, all threads can access every address in the task .

Unlike processes, thread are design to assist one other. But processes might or might not assist one another because processes may originate from different users.

II. Short term and mid term scheduler and long term scheduler

- The **short-term scheduler**, or **CPU Scheduler** – selects job from memory and assigns the CPU to it.
- It must select the new process for CPU frequently.
- Often the short term scheduler executes at least once every 100 milliseconds and it must be fast.

Mid term scheduler:

- The medium-term scheduler temporarily removes processes from main memory and places them in secondary memory (such as a hard disk drive) or vice-versa, which is commonly referred to as "swapping out" or "swapping in".
- To remove process from memory and thus reduce the degree of multiprogramming (number of processes in memory).
- Mid term scheduler makes a proper mix of processes(CPU bound and I/O bound).

Long-term scheduler:

- A long-term scheduler or Job scheduler – selects jobs from the job pool (of secondary memory, disk) and loads them into the memory for execution.

- Long-term scheduler is invoked very infrequently (seconds, minutes)
- The long-term scheduler controls the degree of multiprogramming.

III. User level threads and kernel level threads.

User level threads:

- Each thread is represented simply by a PC, registers, stack and a small control block, all stored in the user process address space.
- Thread management done by user-level threads library. Three primary thread libraries: POSIX Pthreads ,Win32 threads ,Java threads
- User-level threads implement in user-level libraries, rather than via systems calls, so thread switching does not need to call operating system and to cause interrupt to the kernel.
- Fast and Efficient: Thread switching is not much more expensive than a procedure call.
- Simple Management: This simply means that creating a thread, switching between threads and synchronization between threads can all be done without intervention of the kernel.

Kernel level threads:

- Supported by the Kernel.
- Operating Systems kernel provides system call to create and manage threads.
- Instead of thread table in each process, the kernel has a thread table that keeps track of all threads in the system.
- The kernel-level threads are slow and inefficient. For instance, threads operations are hundreds of times slower than that of user-level threads.
- Kernel must manage and schedule threads as well as processes. It requires a full thread control block (TCB) for each thread to maintain information about threads. As a result there is significant overhead and increased in kernel complexity.

4. What is the need for context switching?

- To give each process on a multiprogrammed machine a fair share of the CPU, a hardware clock generates interrupts periodically. This allows the operating system to schedule all processes in main memory (using scheduling algorithm) to run on the CPU at equal intervals.
- Each time a clock interrupt occurs, the interrupt handler checks how much time the current running process has used. If it has used up its entire time slice, then the CPU

scheduling algorithm (in kernel) picks a different process to run. Each switch of the CPU from one process to another is called a context switch.

5. Is CPU scheduling necessary?

   When more than one process is runable, the operating system must decide which one first to be executed. The problem of determining when processors should be assigned and to which processes is called processor scheduling or CPU scheduling. The assignment of physical processors to processes allows processors to accomplish this work.

6. What is process co-operation? Give reasons for process co-operation.

   Processes that can affect other processes or be affected by other processes executing in the system is **process co-operation.**

   Co-operation among processes are allowed for following reasons –

   Information Sharing - There may be several processes which need to access the same file. So the information must be accessible at the same time to all users.

   Computation speedup - Often a solution to a problem can be solved faster if the problem can be broken down into sub-tasks, which are solved simultaneously (particularly when multiple processors are involved.)

   Modularity - A system can be divided into cooperating modules and executed by sending information among one another.

   Convenience - Even a single user can work on multiple task by information sharing.

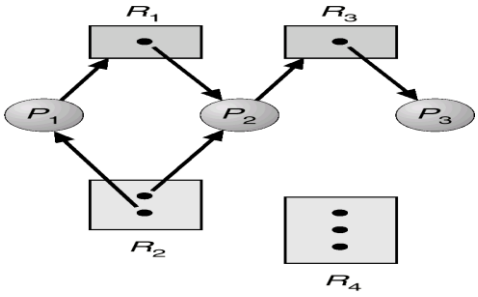
7. Why is a thread called a Light Weight Process?

   A thread is a basic unit of CPU utilization. It consists of a thread ID, own program counter, stack and set of registers, but sharing common code, data, and certain structures such as open files. Such process are called as **lightweight process**.

DEADLOCKS

1. Deadlock exists if a cycle exists. Yes or no. Justify your answer with a suitable example.

   Yes deadlock exists if a cycle exists, if a resource type has single instance.



Example of a Resource Allocation Graph

• **The sets P, *R, and £ :***

P={P1,P2,P3}

R= {R1,R2,R3,R4}

£ = {P1→R1, *P2 →R3, R1* →P2, R2 →P2,R2→P1, R3 →P3}

Resource instances:

• One instance of resource type R1

• Two instances of resource type *R2*

• One instance of resource type *R3*

• Three instances of resource type R4

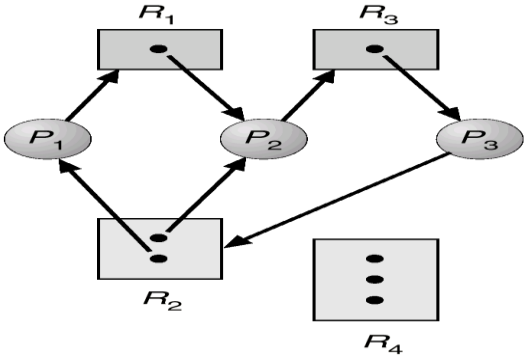
Process states:

• Process P1 is holding an instance of resource type R2 and is waiting for an instance of resource type R1

• Process P2 is holding an instance of R1 and R2 and is waiting for an instance of R3

• Process P3 is holding an instance of R3.

Example of a Resource Allocation Graph with deadlock



Cycles:

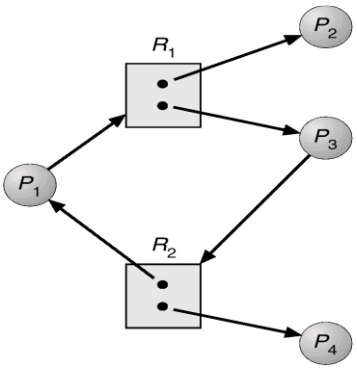
P1 → R1 → P2 → R3 → P3 → R2 → P1

P2 → R3 → P3 → R2 → P2

Processes P1, P2, and P3 are deadlocked.

Process P2 is waiting for resource R3, which is held by P3. Process P3 is waiting for P1 or P2 to release R2 Process P1 is waiting for process P2 to release resource R1

Resource Allocation Graph With A Cycle But No Deadlock



Cycle :
P1→R1 →P3 →R2 →P1

P4 May Release R2 So No Deadlock

• If the graph contains no cycles, then no process in the system is deadlocked

• If the graph does contain a cycle, then a deadlock may exist if only one instance per resource type, then deadlock.

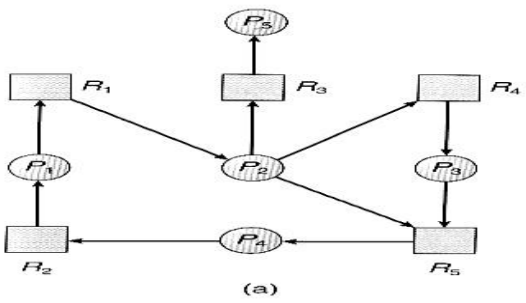
- if several instances per resource type, possibility of deadlock.

2 How wait-for-graph is useful for detection of deadlock?

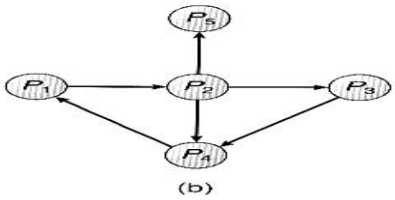
If all the resources have only a single instance then we can define deadlock detection algorithm that uses a variant of resource allocation graph called a **wait for graph**. This graph is obtained by removing the nodes of type resources and removing appropriate edges.

An edge from Pi to Pj in wait for graph implies that Pi is waiting for Pj to release a resource that Pi needs. An edge from Pi to Pj exists in wait for graph if and only if the corresponding resource allocation graph contains the edges Pi → Rq and Rq → Pj.

Deadlock exists within the system if and only if there is a cycle.



Resource-Allocation Graph



Corresponding wait-for graph

Above figures shows resource allocation graph and its wait for graph. In the above wait for graph one cycle exists. Processes involved in the cycle : P1,P2,P3. Therefore a deadlock exists in the system if and only if the wait for graph contains a cycle. To detect deadlocks, the system needs to maintain the wait for for graph and periodically invoke an algorithm that searches for a cycle in the graph.

1. Why is deadlock state more critical than starvation in a multiprogramming environment?

- When processes request a resource and if the resources are not available at that time the process enters into waiting state. Waiting process may not change its state because there sources they are requested are held by other process. This situation is called **deadlock**.
- Deadlocks A group of processes are waiting for resources held by others in the group. None of them will ever make progress
- **Starvation** is the name given to the indefinite postponement of a process because it requires some resource before it can run, but the resource, though available for allocation, is never allocated to this process.
- A process may wait indefinitely because other processes keep coming in and getting the requested resources before this process does. Note that resource is being actively used and the process will stop waiting if other processes stop coming in.

Thus deadlock state more critical than starvation.

MEMORY MANAGEMENT STRATEGIES

1. Explain Belady's anomaly with an example.
2. Explain the concept of forward mapped page table.

   Hierarchical paging:

   • Two-level paging algorithm is used when page table becomes large in which the page table itself is also paged. Eg:-In a 32 bit machine with page size of 4kb. A logical address is divided in to a page number consisting of 20 bits and a page offset of 12 bit. The page table is further divided since the page table is paged, the page number is further divided in to 10 bit page number and a 10 bit offset. So the logical address is as follows:

Page number		page offset
P1	P2	d

   Where p1 is an index into outer page table and p2 is the displacement within the page of the outer page table. The address –translation method for this architecture is shown below.

   Diagram

Because address translation works from outer page table inward, this is known as a forward-mapped page table.
Diagram

3. What do you mean by address binding? Explain with necessary steps, the binding of instructions and data to memory address.

Address Binding:-

• Programs are stored on the secondary storage disks as binary executable files.
• When the programs are to be executed they are brought in to the main memory and placed within a process.
• The collection of processes on the disk waiting to enter the main memory forms the input queue.
• One of the processes which are to be executed is fetched from the queue and placed in the Main memory.
• During the execution it fetches instruction and data from main memory. After the process Terminates it returns back the memory space.
• During execution the process will go through different steps and in each step the address Is represented in different ways.
• In source program the address is symbolic.
• The compiler converts the symbolic address to re-locatable address.
• The loader will convert this re-locatable address to absolute address.

Binding of instructions and data can be done at any step along the way:-

1.Compile time:- If we know whether the process resides in memory then absolute code can be generated. If the static address changes then it is necessary to re-compile the code from the beginning.

2. Load time:- IF THE COMPILER DOESN'T KNOW whether the process resides in memory the and it generates the re-locatable code. In this the binding is delayed until the load time.

3.Execution time:- If the process is moved during its execution from one memory segment to another then the binding is delayed until run time. Special hardware is used for this. Most of the general purpose operating system uses this method.

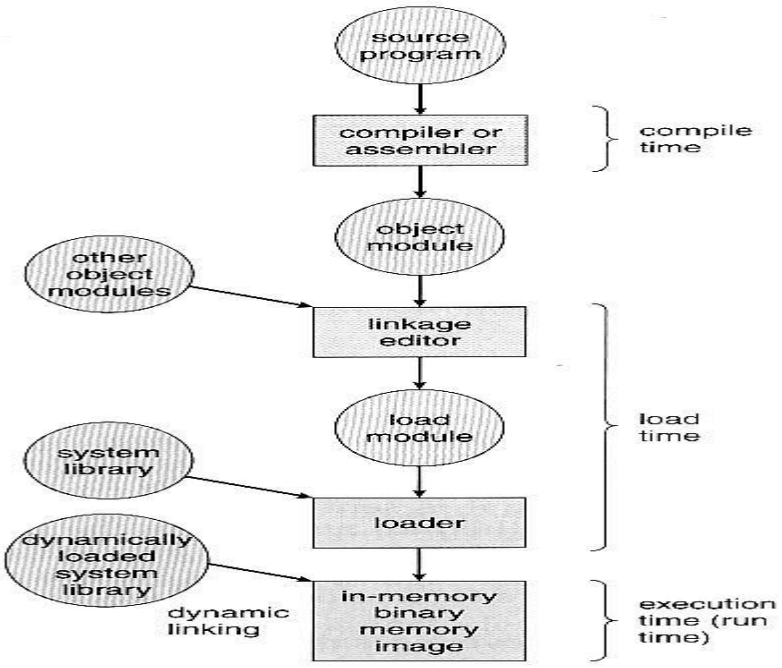


Figure 8.3 Multistep processing of a user program.

4. Explain with illustrations, the internal and external fragmentation problem encountered in contiguous memory allocation.

• Memory fragmentation can be of two types:-

O Internal Fragmentation

O External Fragmentation

• In Internal Fragmentation there is wasted space internal to a portion due to the fact that block ofdata loaded is smaller than the partition.

• *Eg*:- If there is a block of 50kb and if the process requests 40kb and if the block is allocated to the process then there will be 10kb of memory left.

• External Fragmentation exists when there is enough memory space exists to satisfy the request, but it not contiguous i.e., storage is fragmented in to large number of small holes.

• External Fragmentation may be either minor or a major problem.

• One solution for over-coming external fragmentation is compaction. The goal is to move all the free memory together to form a large block. Compaction is not possible always. If the re-location is static and is done at load time then compaction is not possible.

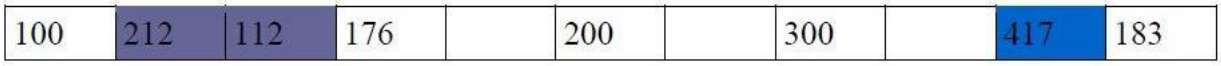
Compaction is possible if the re-location is dynamic and done at execution time.

• Another possible solution to the external fragmentation problem is to permit the logical

Address space of a process to be non-contiguous, thus allowing the process to be allocated physical memory whenever the latter is available.

5. Memory partitions of 100kb,500 kb,200 kb,300kb,600 kb are available how would best worst, first fit algorithm to place processes 212,417,112,426 in order. Which is the best algorithm?

First fit



There is no free space to insert 426.

Best fit

Worst fit

There is no free space to insert 426.

Best fit algorithm is the best algorithm.

6. What is dynamic storage allocation? Explain commonly used strategies for dynamic storage allocation.

- One of the simplest method for memory allocation is to divide memory in to several fixed partition. Each partition contains exactly one process. The OS keeps a table indicating which part of the memory is free and is occupied.
- When a process requests, the OS searches for large hole for this process, hole is a large block of free memory available. If the hole is too large it is split in to two. One part is allocated to the requesting process and other is returned to the set of holes. The set of holes are searched to determine which hole is best to allocate. This procedure is **dynamic storage allocation.**

  There are three strategies to select a free hole:

  o **First bit**:-Allocates first hole that is big enough. This algorithm scans memory from the beginning and selects the first available block that is large enough to hold the process.

o **Best bit**:-It chooses the hole i.e., closest in size to the request. It allocates the smallest hole i.e., big enough to hold the process.

o **Worst fit**:-It allocates the largest hole to the process request. It searches for the largest hole in the entire list.

First fit and best fit are the most popular algorithms for dynamic memory allocation. First fit is generally faster. Best fit searches for the entire list to find the smallest hole i.e., large enough. Worst fit reduces the rate of production of smallest holes.

• All these algorithms suffer from fragmentation.

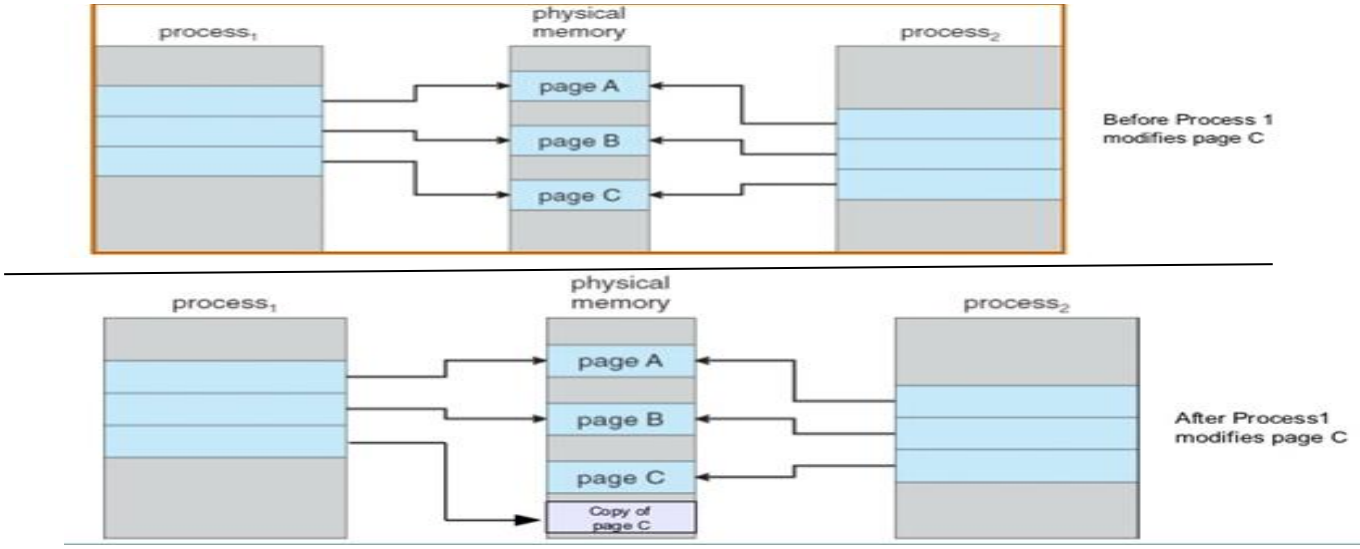
7. **Distinguish between:**

Logical address space and physical address space.

- The address generated by the CPU is called **logical address** or virtual address.
- The address seen by the memory unit i.e., the one loaded in to the memory register is called the **physical address.**
- Compile time and load time address binding methods generate some logical and physical address.
- The execution time addressing binding generate different logical and physical address.
- Set of logical address space generated by the programs is the **logical address space**.
- Set of physical address corresponding to these logical addresses is the **physical address space.**
- The mapping of virtual address to physical address during run time is done by the hardware device called **memory management unit (MMU).**

8. What do you mean by a Copy-on-write? Where it is used? Explain in brief.

Copy-on-write technique initially allows the parent and the child to share the same pages. These pages are marked as copy on-write pages i.e., if either process writes to a shared page, a copy of shared page is created. Eg:-If a child process try to modify a page containing portions of the stack; the OS recognizes them as a copy-on-write page and create a copy of this page and maps it on to the address space of the child process. So the child process will modify its copied page and not the page belonging to parent. The new pages are obtained from the pool of free pages.



9. Differences between segmentation and paging.

   Segmentation:

   - Program is divided in to variable sized segments.
   - User is responsible for dividing the program in to segments.
   - Segmentation is slower than paging.
   - Visible to user. x Eliminates internal fragmentation.
   - Suffers from external fragmentation.
   - Process or user segment number, offset to calculate absolute address.

   Paging:

   - Programs are divided in to fixed size pages.
   - Division is performed by the OS.
   - Paging is faster than segmentation. x Invisible to user.
   - Suffers from internal fragmentation.
   - No external fragmentation.
   - Process or user page number, offset to calculate absolute address.

10. Explain the concept of forward mapped page table.

11. What is locality of reference?

Locality of Reference:

As the process executes it moves from locality to locality. A locality is a set of pages that are actively used.

A program may consist of several different localities, which may overlap.

Locality is caused by loops in code that find to reference arrays and other data structures by indices.

The ordered list of page number accessed by a program is called reference string. Locality is of two types 1) spatial locality 2) temporal locality.