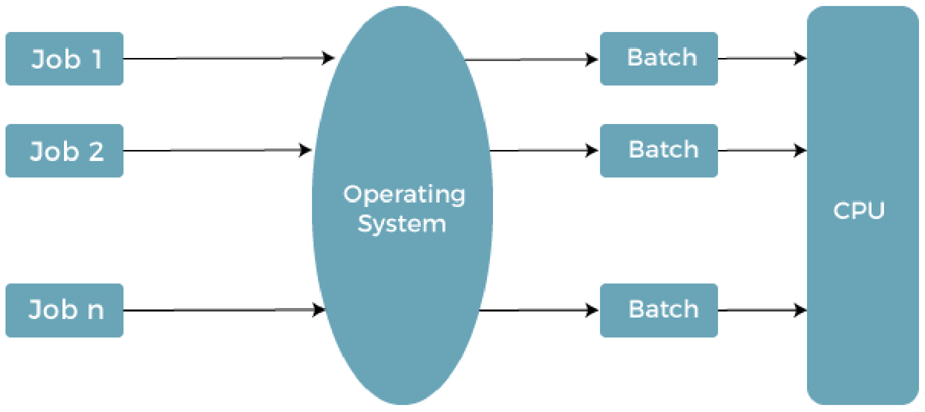
Define batch operating system.

The batch operating system grouped jobs that perform similar functions. These job groups are treated as a batch and executed simultaneously. Users using batch operating systems do not interact directly with the computer. The batch operating system also eliminates the setup time issue.



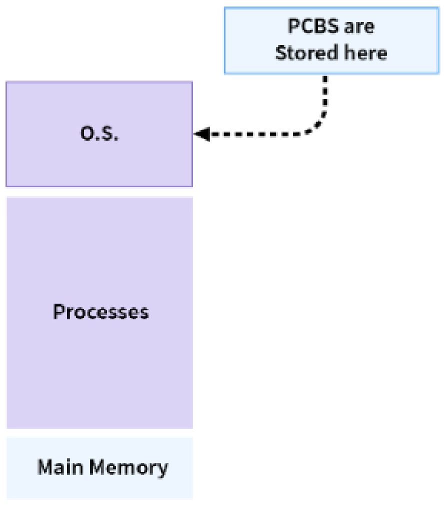
What are the main functions of an Operating System?

Operating System performs memory management, processor management, device management, file management, network management, job accounting and booting. It also acts as an interface between the user and the computer hardware. It also provides security and error-detecting aids.

Define the term PCB.

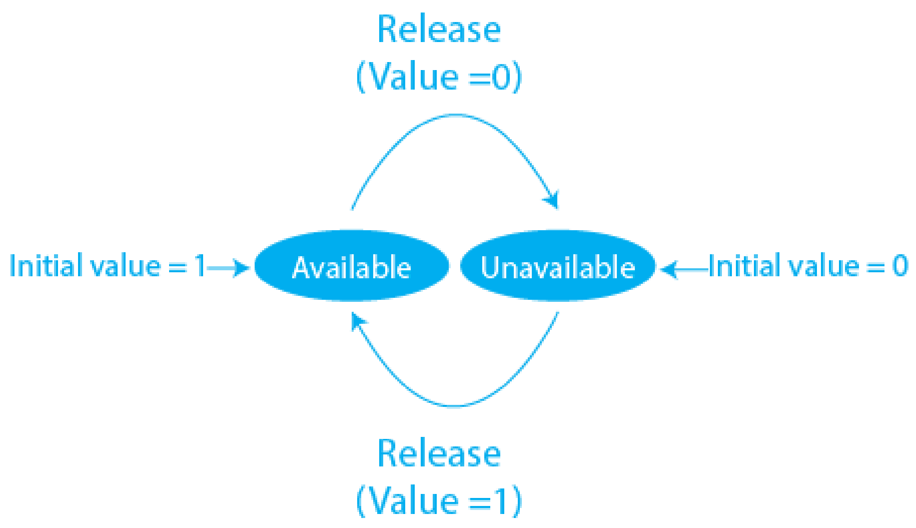
When the process is created by the operating system it creates a data structure to store the information of that process. This is known as Process Control Block (PCB).

Process Control block (PCB) is a data structure that stores information of a process. PCBs are stored in specially reserved memory for the operating system known as kernel space.

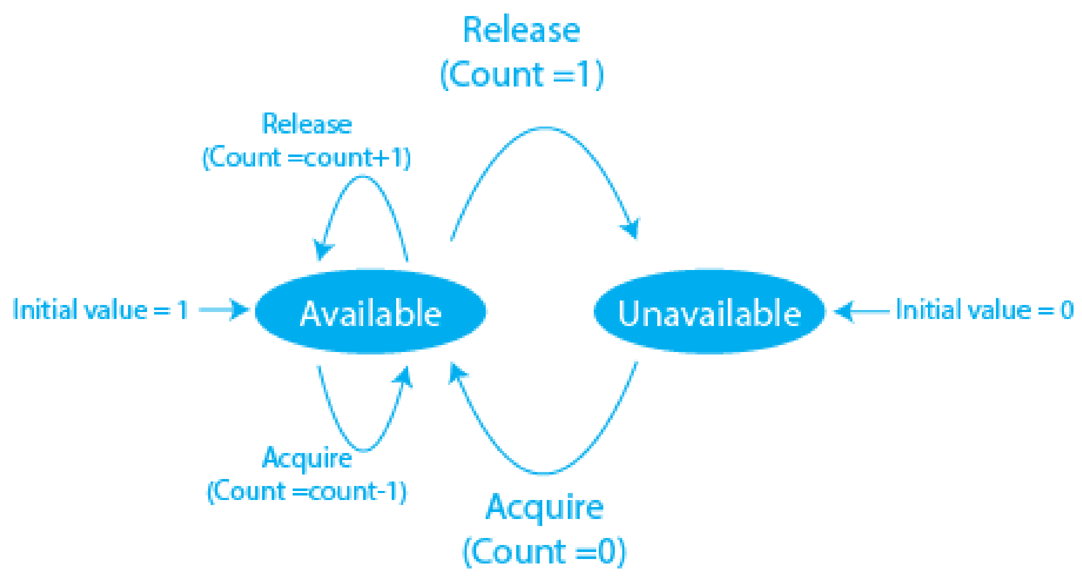


Discuss different types of semaphore.

Binary Semaphores: Binary semaphores are also known as mutual exclusion. A binary semaphore can only have two states: 0 or 1. It is typically used to implement mutual exclusion, which ensures that only one process or thread can access a shared resource at any given time.



Counting Semaphores: Counting semaphores can have any non-negative integer value. They are typically used to manage a pool of resources that can be accessed by multiple processes or threads.



Explain the concept of disk scheduling.

Disk scheduling is done by operating systems to schedule I/O requests arriving for the disk. Disk scheduling is also known as I/O scheduling. Disk scheduling is important because:

- Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by the disk controller. Thus other I/O requests need to wait in the waiting queue and need to be scheduled.
- Two or more requests may be far from each other so can result in greater disk arm movement.
- Hard drives are one of the slowest parts of the computer system and thus need to be accessed in an efficient manner.

What is thrashing? When does it occur?

A process is said to be thrashing if the CPU spends more time serving page faults than executing the pages. This leads to low CPU utilization and the Operating System in return tries to increase the degree of multiprogramming.

Three main causes of thrashing.

1. High degree of Multiprogramming.
2. Less number of frames compared to the processes required.
3. The process scheduling scheme which swaps in more processes when CPU utilization is low.

Write short notes on the following: (a) Kernel (b) Virtual memory (c) Segmentation (d) Distributed System

(a)

Kernel is one of the first programs to be loaded up on the memory before the boot loader. It manages the memory as well as peripherals such as keyboards and monitors.
It acts as a primary interface between the hardware and the processes of a computer. The Kernel connects these two to adjust resources as effectively as possible.

Purpose of Kernel

- It determines the processes that will go next in the CPU (Central Processing Unit), for how long, and when.
- It monitors what amount of memory is being used to store what and where it is stored.
- Kernel serves as an interface between the processes and the hardware.
- It receives requests from the processes through system calls.

(b)

Virtual memory is a part of the system's secondary memory that acts as if it is a part of the main memory. Virtual memory allows a system to execute heavier applications or multiple applications simultaneously without exhausting the RAM (Random Access Memory). In particular, the system can behave as if its total RAM resources were equal to the whole amount of physical RAM plus the complete amount of virtual RAM. When running multiple heavy applications at once, the system's RAM may get overloaded. To mitigate this issue, some data stored in RAM that isn't being actively used can be temporarily relocated to virtual memory. This frees up RAM space, which may then be utilized to store data that the system will need to access.

(c)

X Segmentation divides processes into smaller subparts known as **modules**. The divided segments need not be placed in contiguous memory. Since there is no contiguous memory allocation, internal fragmentation does not take place. The length of the segments of the program and memory is decided by the purpose of the segment in the user program.

There are two types of segmentation:

1. **Virtual memory segmentation –** Each process is divided into a number of segments, but the segmentation is not done all at once. This segmentation may or may not take place at run time of the program.
2. **Simple segmentation –** Each process is divided into a number of segments, all of which are loaded into memory at run time, though not necessarily contiguously.

X (d)
Distributed System is a collection of autonomous computer systems that are physically separated but are connected by a centralized computer network that is equipped with distributed system software. The autonomous computers will communicate among each system by sharing resources and files and performing the tasks assigned to them.

Characteristics of a distributed system are: Resource sharing, Openness, Concurrency, Scalability, Fault tolerance, Transparency and Heterogeneity

Advantages: Resource sharing, Flexibility, Scalability, Fault tolerance and Performance,

Disadvantages: Complexity, Security, Cost and Latency

Applications: Finance and Commerce, Cloud Technologies, Entertainment, Healthcare, Education, Transport and logistics, Environment Management

What are the different states of a process? Explain with the help of diagram. Also explain the concept of process scheduling with queuing.

- **New.** The process is being created.
- **Running.** Instructions are being executed.
- **Waiting.** The process is waiting for some event to occur (such as an I/0 completion or reception of a signal).
- **Ready.** The process is waiting to be assigned to a processor.
- **Terminated.** The process has finished execution.

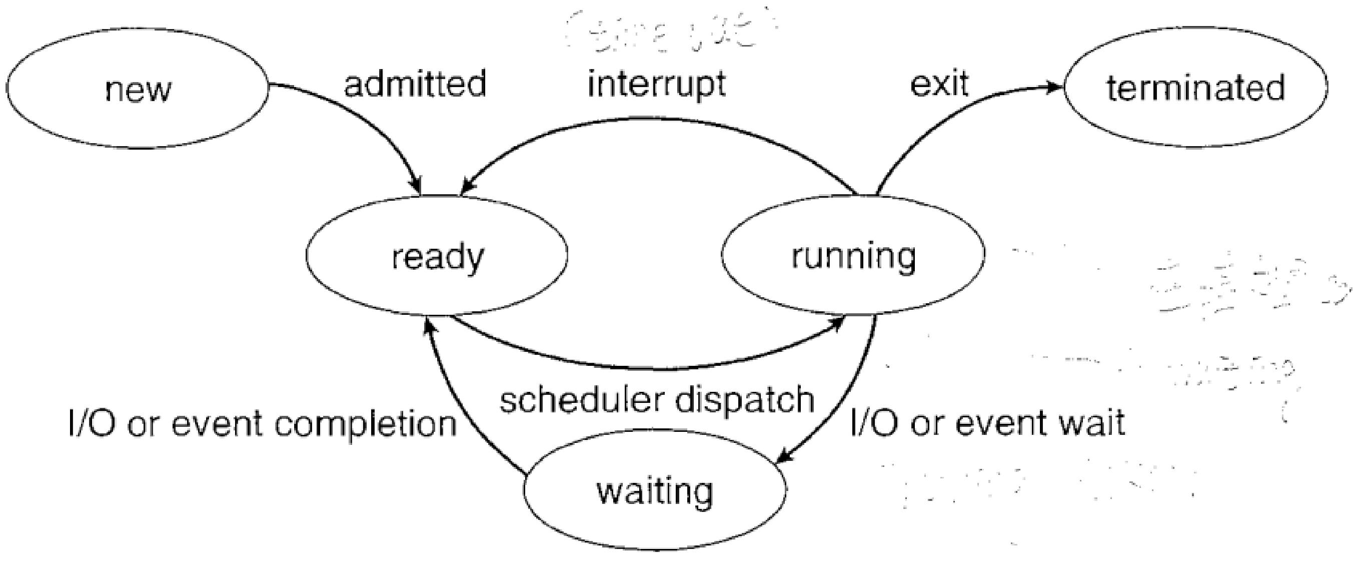


Figure 3.2 Diagram of process state.

Process Scheduling

X The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy. Process scheduling is an essential part of a Multiprogramming operating systems.

There are two categories of scheduling:

1. **Non-preemptive:** Here the resource can't be taken from a process until the process completes execution. The switching of resources occurs when the running process terminates and moves to a waiting state.
2. **Preemptive:** Here the OS allocates the resources to a process for a fixed amount of time. During resource allocation, the process switches from running state to ready state or from waiting state to ready state. This switching occurs as the CPU may give priority to other processes and replace the process with higher priority with the running process.

Process Queues

All PCBs are kept in Process Queues by the OS. The OS has a distinct queue for each process state, and all PCBs in the same execution state are put in the same Process queue. When a process's status changes, its PCB is unbound from the current queue and transferred to the new state queue.
The Operating System manages the following queues:

- **Job queue** − This queue keeps all the processes in the system.
- **Ready queue** − This queue keeps a set of all processes residing in main memory, ready and waiting to execute. A new process is always put in this queue.
- **Device queues** − The processes which are blocked due to unavailability of an I/O device constitute this queue.

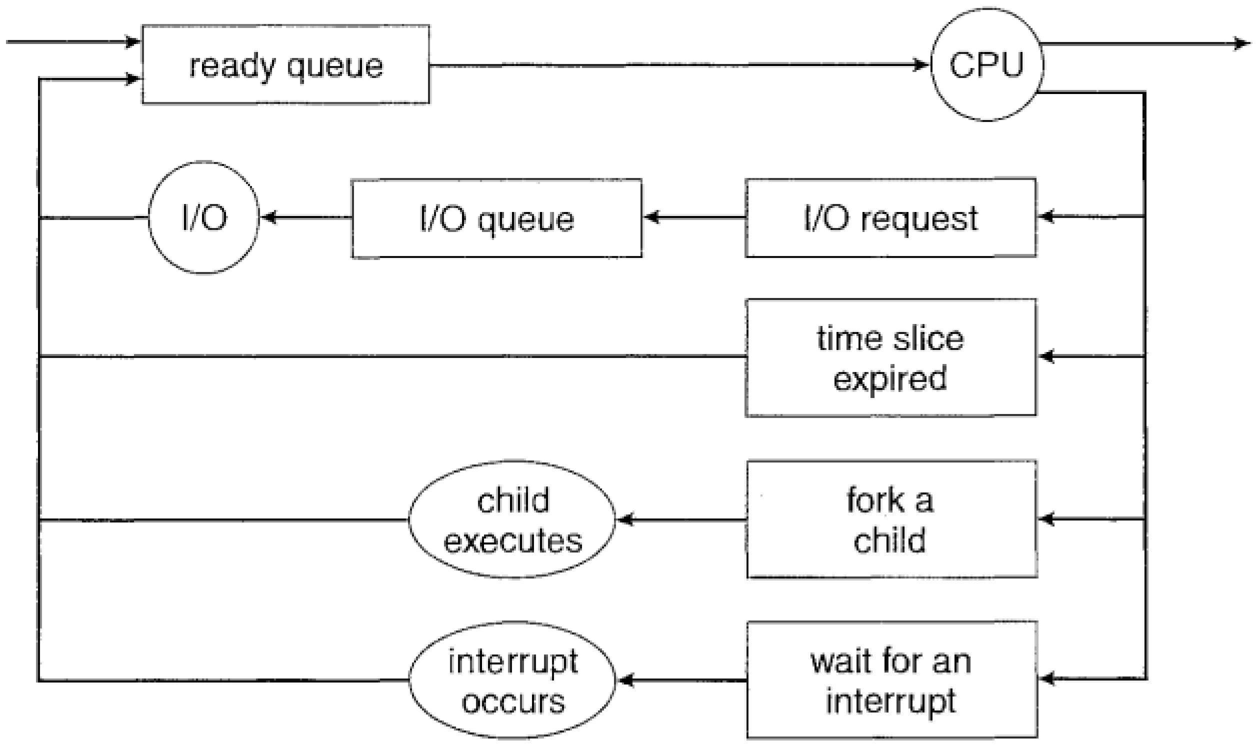


Figure 3.7 Queueing-diagram representation of process scheduling.

What is pre-emptive and non-emptive scheduling? List the CPU scheduling algorithms that support pre-emptive or non-pre-emptive nature of process. Explain any one CPU scheduling algorithm.

PARAMETER	**PREEMPTIVE SCHEDULING**	**NON-PREEMPTIVE SCHEDULING**
Basic	Resources are allocated to a process for a limited time.	Once resources are allocated to a process, the process holds it till it completes its burst time.
Interrupt	Process can be interrupted in between.	Process cannot be interrupted until it terminates itself.
Starvation	If a process having high priority frequently arrives in the ready queue, a low priority process may starve.	If a process with a long burst time is running CPU, then later coming process with less CPU burst time may starve.
CPU Utilization	CPU utilization is high.	CPU utilization is low.
Waiting Time	Waiting time is less.	Waiting time is high.
Response Time	Response time is less.	Response time is high.
Examples	Round Robin and Shortest Remaining Time First.	First Come First Serve and Shortest Job First.

Shortest Job First (SJF) Scheduling

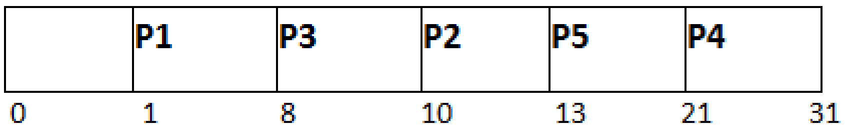
SJF scheduling algorithm, schedules the processes according to their burst time. In SJF scheduling, the process with the lowest burst time, is going to be scheduled next.

Example

PID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
1	1	7	8	7	0
2	3	3	13	10	7

3	6	2	10	4	2
4	7	10	31	24	14
5	9	8	21	12	4

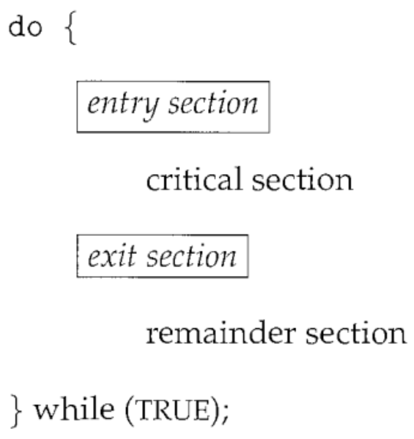
Gantt chart



Explain critical section problem. Give Dining Philosopher problem solution using semaphore.

Critical Section Problem

Consider a system consisting of n processes {P$_0$, P$_1$, ... , P$_{n-1}$}. Each process has a segment of code, called a critical section. No two processes are executing in their critical sections at the same time. The **critical-section problem** is to design a protocol that the processes can use to cooperate. Each process must request permission to enter its critical section. The section of code implementing this request is the entry section. The critical section may be followed by an exit section. The remaining code is the remainder section.



```
do {
        entry section
            critical section
        exit section
            remainder section
} while (TRUE);
```

Figure 6.1 General structure of a typical process P_i.

The solution to the critical section problem must satisfy the following conditions −

- **Mutual Exclusion**: Mutual exclusion implies that only one process can be inside the critical section at any time. If any other processes require the critical section, they must wait until it is free.

- **Progress**: Progress means that if a process is not using the critical section, then it should not stop any other process from accessing it. In other words, any process can enter a critical section if it is free.
- **Bounded Waiting**: Bounded waiting means that each process must have a limited waiting time. It should not wait endlessly to access the critical section.

Dining Philosopher's Problem Solution using Semaphore

Problem Statement: The Dining Philosopher Problem states that K philosophers are seated around a circular table with one chopstick between each pair of philosophers. There is one chopstick between each philosopher. A philosopher may eat if he can pick up the two chopsticks adjacent to him. One chopstick may be picked up by any one of its adjacent followers but not both.

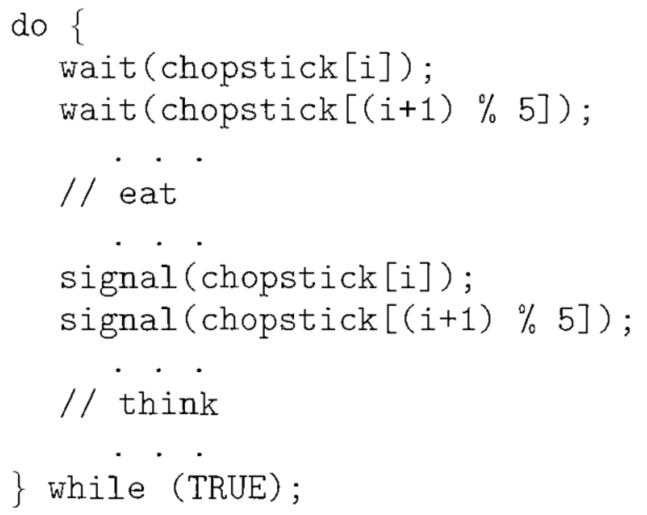
Solution:

A solution of the Dining Philosophers Problem is to use a semaphore to represent a chopstick. A chopstick can be picked up by executing a wait operation on the semaphore and released by executing a signal semaphore.

The structure of the chopstick is shown below –

semaphore chopstick [5];

Initially the elements of the chopstick are initialized to 1 as the chopsticks are on the table and not picked up by a philosopher.



```
do {
   wait(chopstick[i]);
   wait(chopstick[(i+1) % 5]);
      . . .
   // eat
      . . .
   signal(chopstick[i]);
   signal(chopstick[(i+1) % 5]);
      . . .
   // think
      . . .
} while (TRUE);
```

Figure 6.15 The structure of philosopher *i*.

In the above structure, first wait operation is performed on chopstick[i] and chopstick[(i+1) % 5]. This means that the philosopher i has picked up the chopsticks on his sides. Then the eating function is performed.

After that, signal operation is performed on chopstick[i] and chopstick[(i+1) % 5]. This means that the philosopher i has eaten and put down the chopsticks on his sides. Then the philosopher goes back to thinking.

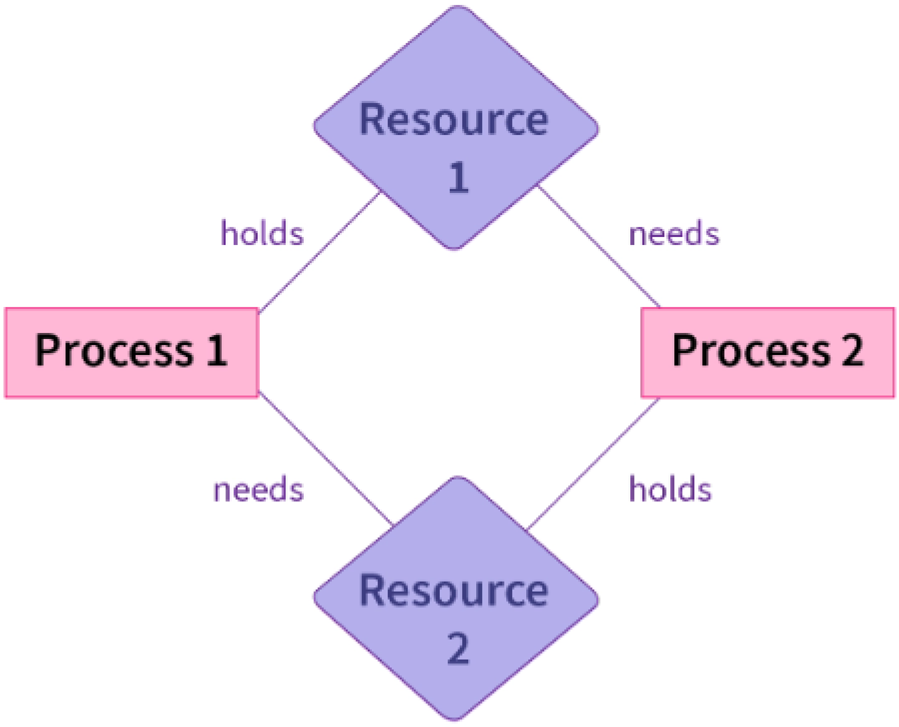
Explain the concept of deadlock. What are the necessary conditions for deadlock occurrence? Discuss with the help of an example.

A **deadlock** is a situation in which more than one process is blocked because it is holding a resource and also requires some resource that is acquired by some other process. Therefore, none of the processes gets executed.

The four necessary conditions for a deadlock to arise are as follows.

- **Mutual Exclusion**: Only one process can use a resource at any given time i.e. the resources are non-sharable.
- **Hold and wait**: A process is holding at least one resource at a time and is waiting to acquire other resources held by some other process.
- **No preemption**: The resource can be released by a process voluntarily i.e. after execution of the process.
- **Circular Wait**: A set of processes are waiting for each other in a circular fashion. For example, let's say there are a set of processes {*P*0, *P*1, *P*2, *P*3} such that *P*0 depends on *P*1, *P*1 depends on *P*2, *P*2 depends on *P*3 and *P*3 depends on *P*0. This creates a circular relation between all these processes and they have to wait forever to be executed.

Example



In the above figure, there are two processes and two resources. Process 1 holds "Resource 1" and needs "Resource 2" while Process 2 holds "Resource 2" and requires "Resource 1". This creates a situation of deadlock because none of the two processes can be executed. Since the resources are non-shareable they can only be used by one process at a time(Mutual Exclusion). Each process is holding a resource and waiting for the other process the release the resource it requires. None of the two processes releases their resources before their execution and this creates a circular wait. Therefore, all four conditions are satisfied.

Explain the term paging in memory management and also discuss which type of fragmentation occurs in paging and how it can be resolved.

Paging is a technique that divides memory into fixed-sized blocks. The main memory is divided into blocks known as Frames and the logical memory is divided into blocks known as Pages. The concept of Paging in OS includes dividing each process in the form of pages of equal size and also, the main memory is divided in the form of frames of fixed size. Now, each page of the process when retrieved into the main memory, is stored in one frame of the memory, and hence, it is also important to have the pages and frames of equal size for mapping and maximum utilization of the memory.

When we use a paging scheme, we may have some internal fragmentation.

If the memory requirements of a process do not coincide with page boundaries, the last frame allocated may not be completely full, resulting in internal fragmentation.

It may be solved by assigning space to the process via dynamic partitioning. Dynamic partitioning allocates only the amount of space requested by the process. As a result, there is no internal fragmentation.

Explain the concept of a file and file operations. What are the different access methods in file system?

A file is a collection of related data stored on the secondary storage in a sequence of operations. The content of the files is defined by its creator who is creating the file. The various operations which can be implemented on a file are called file operations. Some common operations are as follows:

1.Create operation: This operation is used to create a file in the file system. The file system allocates space to the file and entry of this new file is made into the appropriate directory.

2. Open operation: When the user wants to open a file, it provides a file name to open the particular file in the file system. It tells the operating system to invoke the open system call and passes the file name to the file system.

3. Write operation: This operation is used to write the information into a file. A write system call is issued that specifies the name of the file and the length of the data has to be written to the file.

4. Read operation: This operation reads the contents from a file. A Read pointer is maintained by the OS, pointing to the position up to which the data has been read.

5. Re-position or Seek operation: The seek system call, re-positions the file pointers from the current position to a specific place in the file.

6. Delete operation: All the associated file space and the directory entry is released.

7. Truncate operation: Truncating is simply deleting the file except deleting attributes.

8. Close operation: On closing the file, all the internal descriptors are deallocated and all the occupied resources are released.

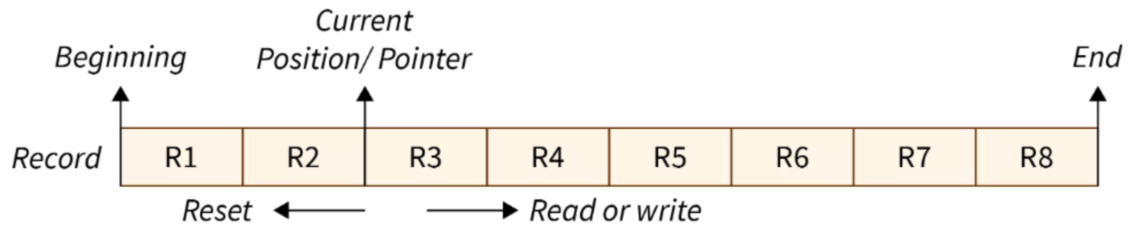
9. Append operation: This operation adds data to the end of the file.

10. Rename operation: This operation is used to rename the existing file.

File Access Methods:

Sequential Access

The operating system reads the file word by word in sequential access method of file accessing. A pointer is made, which first links to the file's base address. If the user wishes to read the first word of the file, the pointer gives it to them and raises its value to the next word. This procedure continues till the file is finished. It is the most basic way of file access. The data in the file is evaluated in the order that it appears in the file



Sequential Access Mechanism

Advantages of Sequential Access:

- The sequential access mechanism is very easy to implement.
- It uses lexicographic order to enable quick access to the next entry.

Disadvantages of Sequential Access:

- Sequential access will become slow if the next file record to be retrieved is not present next to the currently pointed record.
- Adding a new record may need relocating a significant number of records of the file.

Direct Access

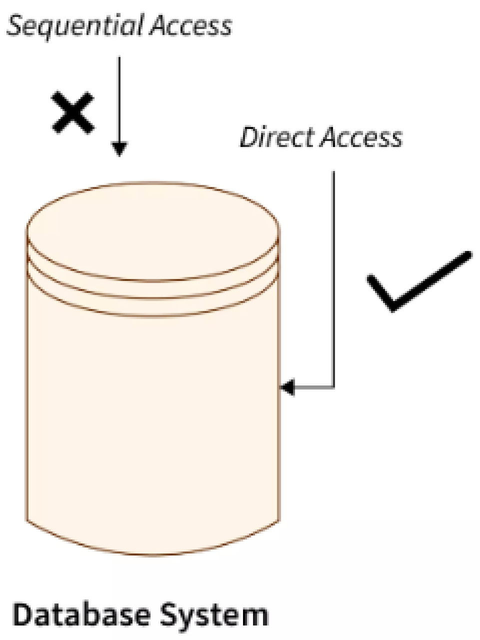
Another method is *direct access method* also known as *relative access method*. A file is made up of fixed-length logical record that allows the program to read and write record rapidly. The direct access is based on the disk model of a file since disk allows random access to any file block. For direct access, the file is viewed as a numbered sequence of block or record. There is no restriction on the order of reading and writing for a direct access file. A block number provided by the user to the operating system is normally a *relative block number*, the first relative block of the file is 0 and then 1 and so on.

Advantages of Direct/Relative Access:

- The files can be retrieved right away with direct access mechanism, reducing the average access time of a file.
- There is no need to traverse all of the blocks that come before the required block to access the record.

Disadvantages of Direct/Relative Access:

- The direct access mechanism is typically difficult to implement due to its complexity.
- Organizations can face security issues



Index access method

It is constructed on top of the sequential access mechanism. In this method, there is an index that holds the pointers to various blocks of the file. In order to access any block of the file, one has to first access the index, and from there, we can get the pointers to various blocks. It is a modification of the sequential access method. It allows random access. Apart from the file records, an extra index is required to keep track of the blocks.

Advantages of Indexed Sequential Access:

- If the index table is appropriately arranged, it accesses the records very quickly.
- Records can be added at any position in the file quickly.

Disadvantages of Indexed Sequential Access:

- When compared to other file access methods, it is costly and less efficient.
- It needs additional storage space.

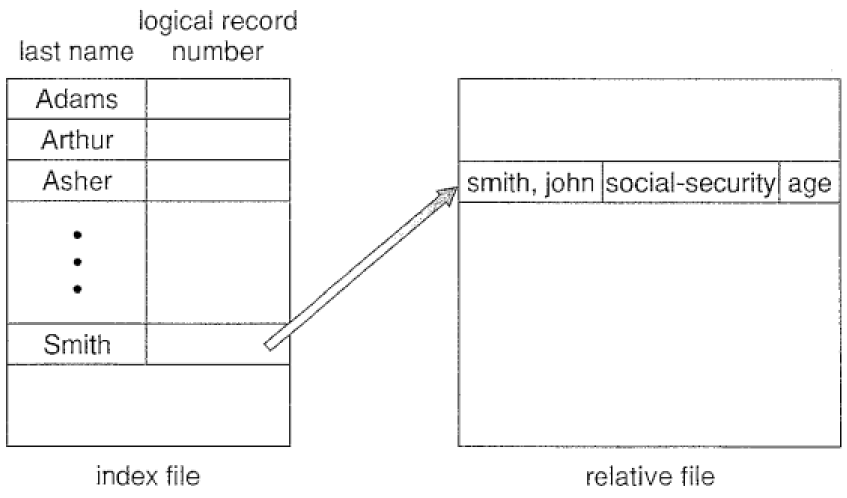


Figure 10.5 Example of index and relative files.

What are the different page replacement algorithms? Which algorithm suffers from Belady's anomaly and why. Explain with a suitable example.

https://www.scaler.com/topics/operating-system/page-replacement-algorithm/

What are disk scheduling algorithms? Why these algorithms are not implemented in hardware, instead of Operating System Implementation?

https://www.geeksforgeeks.org/disk-scheduling-algorithms/

Disk scheduling algorithms are not implemented in hardware, instead of operating system implementation because software is a more flexible and cost-effective platform for implementing these algorithms, and the performance benefits of implementing them in hardware are not significant.