MUST-DO Questions for Interviews (DBMS, CN and OS)

SDE-Sheet(Core) by Take U Forward (Striver_79)
Disclaimer: All the below solutions are directly collected from trustable resources like wikipedia , geeksforgeeks, javatpoint, tutorialspoint , guru99 , Apni kaksha notes, etc.. Sole purpose of sharing this is to help people who don't have enough time to collect answers for every question.
Thank you Striver for this question bank!

Operating Systems

1. What is the main purpose of an operating system? Discuss different types?

   An operating system (OS) is system software that manages computer hardware, software resources, and provides common services for computer programs.
   So it manages the computer's memory, processes, devices, files, and security aspects of the system. It also allows us to communicate with the computer without knowing how to speak the computer's language. Without an operating system, a computer is not useful.

   Types of Operating Systems:
   https://www.geeksforgeeks.org/types-of-operating-systems/

   1. **Batch OS** (eg: payroll system, bank statements, data entry, etc.)
      Batch OS is the first operating system for second-generation computers. This OS does not directly interact with the computer. Instead, an operator takes up similar jobs and groups them together into a batch, and then these batches are executed one by one based on the first-come, first, serve principle.

   2. **Distributed OS** (eg: LOCUS)
      In a distributed OS, various computers are connected through a single communication channel. These independent computers have their memory unit and CPU and are known as loosely coupled systems.Failure of one system will not affect the other systems because all the computers are independent of each other.

   3. **Multitasking OS** (eg: UNIX)
      The multitasking OS is also known as the time-sharing operating system as each task is given some time so that all the tasks work efficiently.Each task gets equal time for execution. The idle time for the CPU will be the lowest.

   4. **Network OS** (eg: Microsoft Windows server 2008, LINUX, etc.)
      Network operating systems are the systems that run on a server and manage all the networking functions. They allow sharing of various files, applications,

printers, security, and other networking functions over a small network of computers like LAN or any other private network.

5. **Real-Time OS** (eg: Medical imaging systems, robots,)
   These operating systems are useful where many events occur in a short time or certain deadlines, such as real-time simulations. (More details in Q.15)

6. **Mobile OS** (eg. Android OS, ios)
   A mobile OS is an operating system for smartphones, tablets, and PDA's. It is a platform on which other applications can run on mobile devices.

2. What is a socket, kernel and monolithic kernel ?

Socket:
A socket is defined as an endpoint for communication, A pair of processes communicating over a network employ a pair of sockets ,one for each process. A socket is identified by an IP address concatenated with a port number.
The server waits for incoming client requests by listening to a specified port. Once a request is received, the server accepts a connection from the client socket to complete the connection.

Kernel is the central core component of an operating system that manages operations of computer and hardware. Kernel

- Establishes communication between user level application and hardware.
- Manages memory and CPU time
- Decides state of incoming processes.
- Controls Disk, Memory, Task Management

Types of kernels:

- ❖ Monolithic Kernel (provides good performance but lots of lines of code)
  It is one of the types of kernel where all operating system services operate in kernel space. It has dependencies between system components. It has huge lines of code which is complex.
  Example : Unix, Linux, Open VMS, XTS-400 etc.

(few more types includes : Micro Kernel (more stable but lots of system calls,context switch), Hybrid Kernel(but this is still similar to monolithic kernel), Exo kernel(fewest hardware abstractions hence more work for app developers))

3. Difference between process and program and thread? Different types of process.

Program	Process	Thread
Program is a set of instructions to perform a certain task. Eg: chrome.exe, notepad.exe	Process is an instance of an executing program. For example, we write our computer programs in a text file and when we execute this program, it becomes a process which performs all the tasks mentioned in the program.	Thread is a path of execution within a process. A thread is also known as a lightweight process. The idea is to achieve parallelism by dividing a process into multiple threads. For example,Word processor uses multiple threads: one thread to format the text, another thread to process inputs,
Program is a passive entity as it resides in the secondary memory	Process is a active entity as it is created during execution and loaded into the main memory	Thread is also active entity as it executes a part of a process
Program exists at a single place and continues to exist until its deleted	Process exists for a limited span of time as it gets terminated after the completion of task	A thread goes through various stages in its lifecycle. For example, a thread is born, started, runs, and then dies.
Program does not have its own control block	Process has its own control block called Process Control Block, Stack and Address Space,	Thread has Parents' PCB, its own Thread Control Block and Stack and common Address space.
	Process run in separate memory space	Threads within the process run in a shared memory space.
	Process switching needs interaction with the operating system.	Thread switching does not need to interact with operating system , hence lesser context switching time as well
	Process is heavy weight or resource intensive.	Thread is lightweight, taking fewer resources than a process.
		Threads are implemented in following two ways −

		User Level Threads − User managed threads. Kernel Level Threads − Operating System managed threads acting on kernel, an operating system core.

4. Define virtual memory, thrashing, threads.

Virtual Memory:
A computer can address more memory than the amount physically installed on the system. This extra memory is actually called virtual memory and it is a section of a hard disk that's set up to emulate the computer's RAM.
The main visible advantage of this scheme is that programs can be larger than physical memory. Virtual memory serves two purposes. First, it allows us to extend the use of physical memory by using a disk. Second, it allows us to have memory protection, because each virtual address is translated to a physical address.

Thrashing:
Thrashing is a condition or a situation when the system is spending a major portion of its time in servicing the page faults, but the actual processing done is very negligible. High degree of multiprogramming(if number of processes keeps on increasing in the memory), lack of frames (if a process is allocated too few frames, then there will be too many and too frequent page faults) causes Thrashing.

Threads:
A thread is a single sequential flow of execution of tasks of a process so it is also known as thread of execution or thread of control.

5. What is RAID ? Different types.

RAID mode	Description	Operation	Advantages	Disadvantages	Recovery
RAID 0	Striped disks	Data is split evenly between two or more disks.	Large size and the fastest speed.	No redundancy.	If one or more drives fails, this results in array failure.
RAID 1	Mirrored disks	Two or more drives have identical data on them.	A single drive failure will not result in data loss.	Speed and size is limited by the slowest and smallest disk.	Only one drive is needed for recovery.
RAID 3	Striped set with dedicated parity	Data is split evenly between two or more disks, plus a dedicated drive for parity storage	High speeds for sequential read/write operations.	Poor performance for multiple simultaneous instructions.	A single drive failure will rebuild.

RAID, or “Redundant Arrays of Independent Disks” is a technique which makes use of a combination of multiple disks instead of using a single disk for increased performance, data redundancy or both.Data redundancy, although taking up extra space, adds to disk reliability. This means, in case of disk failure, if the same data is also backed up onto another disk, we can retrieve the data and go on with the operation. Here are the important RAID types:

Few other RAID types: JBOD, Clone, Big

6. What is a deadlock ? Different conditions to achieve a deadlock.

A Deadlock is a situation where each of the computer processes waits for a resource which is being assigned to some other process. In this situation, none of the processes gets executed since the resource it needs is held by some other process which is also waiting for some other resource to be released.
Deadlock is an infinite waiting.

How deadlock is achieved:
Deadlock happens when Mutual exclusion, hold and wait, No preemption and circular wait occurs simultaneously.

Mutual Exclusion:
A resource can only be shared in a mutually exclusive manner. It implies that two processes cannot use the same resource at the same time.

Hold and Wait:
A process waits for some resources while holding another resource at the same time.

No preemption:
The process which once scheduled will be executed till the completion. No other process can be scheduled by the scheduler meanwhile.

Circular Wait:
All the processes must be waiting for the resources in a cyclic manner so that the last process is waiting for the resource which is being held by the first process.

7. What is fragmentation? Types of fragmentation.

Fragmentation:
An unwanted problem in the operating system in which the processes are loaded and unloaded from memory, and free memory space is fragmented. Processes can't be assigned to memory blocks due to their small size, and the memory blocks stay unused. It is also necessary to understand that as programs are loaded and deleted from memory, they generate free space or a hole in the memory. These small blocks cannot be allotted to new arriving processes, resulting in inefficient memory use.

The conditions of fragmentation depend on the memory allocation system. As the process is loaded and unloaded from memory, these areas are fragmented into small pieces of memory that cannot be allocated to incoming processes. It is called fragmentation.

Causes of Fragmentation:
User processes are loaded and unloaded from the main memory, and processes are kept in memory blocks in the main memory. Many spaces remain after process loading and swapping that another process cannot load due to their size. Main memory is available, but its space is insufficient to load another process because of the dynamical allocation of main memory processes

Types of fragmentation :

Internal Fragmentation
When a process is allocated to a memory block, and if the process is smaller than the amount of memory requested, a free space is created in the given memory block. Due to this, the free space of the memory block is unused, which causes internal fragmentation.
Let's suppose a process P1 with a size of 3MB arrives and is given a memory block of 4MB. As a result, the 1MB of free space in this block is unused and cannot be used to allocate memory to another process. It is known as internal fragmentation.

Solution to internal fragmentation: Dynamic Partitioning

External Fragmentation
External fragmentation happens when a dynamic memory allocation method allocates some memory but leaves a small amount of memory unusable. The quantity of available memory is substantially reduced if there is too much external fragmentation. There is enough memory space to complete a request, but it is not contiguous. It's known as external fragmentation

One of the solution to external fragmentation : Compaction : combining all the free memory into a single large block

8. What is spooling ?

SPOOL is an acronym for simultaneous peripheral operations online.
Spooling is a process in which data is temporarily held to be used and executed by a device, program, or system.
In spooling, there is no interaction between the I/O devices and the CPU. That means there is no need for the CPU to wait for the I/O operations to take place. Such operations take a long time to finish executing, so the CPU will not wait for them to finish.
The biggest example of Spooling is printing. The documents which are to be printed are stored in the SPOOL and then added to the queue for printing. During this time,

many processes can perform their operations and use the CPU without waiting while the printer executes the printing process on the documents one-by-one.

9. What is semaphore and mutex (Differences might be asked)? Define Binary semaphore.

Mutex and Semaphore both provide synchronisation services.

Mutex is a mutual exclusion object that synchronises access to a resource. It is created with a unique name at the start of a program. The Mutex is a locking mechanism that makes sure only one thread can acquire the Mutex at a time and enter the critical section. This thread only releases the Mutex when it exits the critical section.

```
wait (mutex);
…..
Critical Section
…..
signal (mutex);
```

```
A Mutex is different from a semaphore as it is a locking mechanism while a semaphore is a signalling mechanism. A binary semaphore can be used as a Mutex but a Mutex can never be used as a semaphore.
```

```
------------
```

Semaphore: A semaphore is a signalling mechanism and a thread that is waiting on a semaphore can be signalled by another thread. This is different from a mutex as the mutex can be signalled only by the thread that called the wait function.A semaphore uses two atomic operations, wait and signal for process synchronisation.

A semaphore restricts the number of simultaneous users of a shared resource upto a maximum number. Threads can request access to the resource (decrementing the semaphore), and can signal that they have finished using the resource (incrementing the semaphore).

The wait operation decrements the value of its argument S, if it is positive. If S is negative or zero, then no operation is performed.

```
wait(S)
{
  while (S<=0);
  S--;
}
```

The signal operation increments the value of its argument S.

```
signal(S)
{
```

```
  S++;
}
```

There are mainly two types of semaphores i.e. counting semaphores and binary semaphores.

Counting Semaphores are integer value semaphores and have an unrestricted value domain. These semaphores are used to coordinate the resource access, where the semaphore count is the number of available resources.

The binary semaphores are like counting semaphores but their value is restricted to 0 and 1. The wait operation only works when the semaphore is 1 and the signal operation succeeds when semaphore is 0.

10. Belady's Anomaly

Bélády's anomaly is the name given to the phenomenon where increasing the number of page frames results in an increase in the number of page faults for a given memory access pattern.

Solution to fix Belady's Anomaly:
Implementing alternative page replacement algo helps eliminate Belady's Anomaly..
Use of stack based algorithms, such as Optimal Page Replacement Algorithm and Least Recently Used (LRU) algorithm, can eliminate the issue of increased page faults as these algorithms assign priority to pages.

11. Starving and Ageing in OS

Starving/Starvation(also called Lived lock):
Starvation is the problem that occurs when low priority processes get jammed for an unspecified time as the high priority processes keep executing. So starvation happens if a method is indefinitely delayed.

Solution to Starvation : Ageing

Ageing is a technique of gradually increasing the priority of processes that wait in the system for a long time.

12. Why does thrashing occur?

High degree of multiprogramming(if number of processes keeps on increasing in the memory) , lack of frames(if a process is allocated too few frames, then there will be too many and too frequent page faults.) causes Thrashing.

13. What is paging and why do we need it?

Paging is a memory management scheme that eliminates the need for contiguous allocation of physical memory. This scheme permits the physical address space of a process to be non – contiguous.
Paging is used for faster access to data. When a program needs a page, it is available in the main memory(RAM) as the OS copies a certain number of pages from your storage device to main memory. Paging allows the physical address space of a process to be noncontiguous.

14. Demand Paging, Segmentation

Demand paging is a method of virtual memory management which is based on the principle that pages should only be brought into memory if the executing process demands them. This is often referred to as lazy evaluation as only those pages demanded by the process are swapped from secondary storage to main memory. So demand paging works opposite to the principle of loading all pages immediately.

Segmentation is a memory management technique in which the memory is divided into the variable size parts. Each part is known as a segment which can be allocated to a process.

The details about each segment are stored in a table called a segment table. Segment table is stored in one (or many) of the segments.

Segment table contains mainly two information about segment:

Base: It is the base address of the segment
Limit: It is the length of the segment.

15. Real Time Operating System, types of RTOS.

A real-time operating system (RTOS) is a special-purpose operating system used in computers that has strict time constraints for any job to be performed and is intended to serve real time applications that possess data as it comes in , typically without buffer delays.

Types of RTOS:

Hard Real Time:
In Hard RTOS, the deadline is handled very strictly which means that a given task must start executing on specified scheduled time, and must be completed within the assigned time duration.

Example: Medical critical care system, Aircraft systems, etc.

Firm Real Time:
These types of RTOS also need to follow the deadlines. However, missing a deadline may not have a big impact but could cause undesired effects, like a huge reduction in quality of a product.

Example: Various types of Multimedia applications.

Soft Real Time:
Soft Real time RTOS, accepts some delays by the Operating system. In this type of RTOS, there is a deadline assigned for a specific job, but a delay for a small amount of time is acceptable. So, deadlines are handled softly by this type of RTOS.

Example: Online Transaction system and Livestock price quotation System.

16. Difference between main memory and secondary memory.

Parameter	Primary Memory	Secondary Memory
Nature	The primary memory is categorized as volatile & nonvolatile memories.	The secondary memory is always a non-volatile memory.
Alias	These memories are also called internal memory.	Secondary memory is known as a Backup memory or Additional memory or Auxiliary memory.
Access	Data is directly accessed by the processing unit.	Data cannot be accessed directly by the processor. It is first copied from secondary memory to primary memory. Only then CPU can access it.
Formation	It's a volatile memory meaning data cannot be retained in case of power failure.	It's a non-volatile memory so that that data can be retained even after power failure.
Storage	It holds data or information that is currently being used by the processing unit. Capacity is usually in 16 to 32 GB	It stores a substantial amount of data and information. Capacity is generally from 200GB to terabytes.
Accesses	Primary memory can be accessed by the data bus.	Secondary memory is accessed by I/O channels.
Expense	Primary memory is costlier than secondary memory.	Secondary memory is cheaper than primary memory.

17. Dynamic Binding

Static binding happens when the code is compiled, while dynamic bind happens when the code is executed at run time.

Static Binding:
When a compiler acknowledges all the information required to call a function or all the values of the variables during compile time, it is called “static binding”. As all the required information is known before runtime, it increases the program efficiency and it also enhances the speed of execution of a program. Static Binding makes a program very efficient, but it declines the program flexibility, as ‘values of variable’ and ‘function calling’ are predefined in the program. Static binding is implemented in a program at the time of coding. Overloading a function or an operator is the example of compile time polymorphism i.e. static binding.

Dynamic Binding Calling a function or assigning a value to a variable, at run-time is called “Dynamic Binding”. Dynamic binding can be associated with run time ‘polymorphism’ and ‘inheritance’ in OOP. Dynamic binding makes the execution of a program flexible as it can decide what value should be assigned to the variable and which function should be called, at the time of program execution. But as this information is provided at run time it makes the execution slower as compared to static binding.

18. FCFS Scheduling and SJF Scheduling

First Come First Served (FCFS)	Shortest Job First (SJF)
First Come First Served (FCFS) executes the processes in the order in which they arrive i.e. the process that arrives first is executed first.	Shortest Job First (SJF) executes the processes based upon their burst time i.e. in ascending order of their burst times.
FCFS is non preemptive in nature.	SJF is also non-preemptive but its preemptive version is also there called Shortest Remaining Time First (SRTF) algorithm.
FCFS results in quite long waiting time for the processes and thus increases average waiting time.	The average waiting time for given set of processes is minimum.
FCFS leads to the convoy effect.	It does not lead to the convoy effect.
FCFS algorithm is the easiest to implement in any system.	The real difficulty with SJF is knowing the length of the next CPU request or burst.
A process may have to wait for quite long to get executed depending on the burst time of the processes that have arrived first.	A long process may never get executed and the system may keep executing the short processes.
FCFS lead to lower device and CPU utilization thereby decreasing the efficiency of the system.	SJF leads to higher effectiveness of the system due to lower average waiting time.
FCFS results in minimal overhead.	In case of SJF, elapsed time should be recorded, results in more overhead on the processor.

19. SRTF Scheduling

SRTF Scheduling is a preemptive version of SJF scheduling. In SRTF, the execution of the process can be stopped after a certain amount of time. At the arrival of every process, the short term scheduler schedules the process with the least remaining burst time among the list of available processes and the running process.

20. LRTF Scheduling

This is a preemptive version of Longest Job First (LJF) scheduling algorithm. In this scheduling algorithm, we find the process with the maximum remaining time and then process it. We check for the maximum remaining time after some interval of time(say 1 unit each) to check if another process having more Burst Time arrived up to that time.

21. Priority Scheduling

Priority Scheduling is a method of scheduling processes that is based on priority. In this algorithm, the scheduler selects the tasks to work as per the priority.

The processes with higher priority should be carried out first, whereas jobs with equal priorities are carried out on a round-robin or FCFS basis. Priority depends upon memory requirements, time requirements, etc.

Priority scheduling divided into two main types:

Preemptive Scheduling
In Preemptive Scheduling, the tasks are mostly assigned with their priorities. Sometimes it is important to run a task with a higher priority before another lower priority task, even if the lower priority task is still running. The lower priority task holds for some time and resumes when the higher priority task finishes its execution.

Non-Preemptive Scheduling
In this type of scheduling method, the CPU has been allocated to a specific process. The process that keeps the CPU busy, will release the CPU either by switching context or terminating. It is the only method that can be used for various hardware platforms. That’s because it doesn’t need special hardware (for example, a timer) like preemptive scheduling.

22. Round Robin Scheduling

In Round-robin scheduling, each ready task runs turn by turn only in a cyclic queue for a limited time slice. This algorithm also offers starvation free execution of processes. Widely used preemptive scheduling method in traditional OS. All the jobs get a fair allocation of CPU. Cons include : Finding a correct time quantum is a quite

difficult task in this system, Round-robin scheduling doesn’t give special priority to more important tasks.

23. Producer Consumer Problem

About Producer-Consumer problem
The Producer-Consumer problem is a classic problem that is used for multi-process synchronisation i.e. synchronisation between more than one processes.

The job of the Producer is to generate the data, put it into the buffer, and again start generating data. While the job of the Consumer is to consume the data from the buffer.

What's the problem here?
The following are the problems that might occur in the Producer-Consumer:

The producer should produce data only when the buffer is not full. If the buffer is full, then the producer shouldn't be allowed to put any data into the buffer.
The consumer should consume data only when the buffer is not empty. If the buffer is empty, then the consumer shouldn't be allowed to take any data from the buffer.
The producer and consumer should not access the buffer at the same time.

We can solve this problem by using semaphores.

24. Banker’s Algorithm

It is a banker algorithm used to avoid deadlock and allocate resources safely to each process in the computer system. The 'S-State' examines all possible tests or activities before deciding whether the allocation should be allowed to each process. It also helps the operating system to successfully share the resources between all the processes. The banker's algorithm is named because it checks whether a person should be sanctioned a loan amount or not to help the bank system safely simulate allocation resources.

25. Explain Cache

Cache memory is an extremely fast memory type that acts as a buffer between RAM and the CPU. It holds frequently requested data and instructions so that they are immediately available to the CPU when needed.

26. Diff between direct mapping and associative mapping

In Cache memory, data is transferred as a block from primary memory to cache memory. This process is known as Cache Mapping.
Direct Mapping

In direct mapping cache, instead of storing total address information with data in cache only part of address bits is stored along with data.

The new data has to be stored only in a specified cache location as per the mapping rule for direct mapping. So it doesn't need a replacement algorithm.

Advantages of direct mapping

Direct mapping is the simplest type of cache memory mapping.
Here only tag fields are required to match while searching words that is why it is the fastest cache.
Direct mapping cache is less expensive compared to associative cache mapping.
Disadvantages of direct mapping

The performance of direct mapping cache is not good as requires replacement for data-tag value.

In associative mapping both the address and data of the memory word are stored.

The associative mapping method used by cache memory is a very flexible one as well as very fast.

This mapping method is also known as a fully associative cache.

Advantages of associative mapping

Associative mapping is fast.
Associative mapping is easy to implement.
Disadvantages of associative mapping

Cache Memory implementing associative mapping is expensive as it requires storing addresses along with the data.

27. Diff between multitasking and multiprocessing

S No.	Multi-tasking	Multiprocessing
1.	The execution of more than one task simultaneously is known as multitasking.	The availability of more than one processor per system, that can execute several set of instructions in parallel is known as multiprocessing.
2.	The number of CPU is one.	The number of CPUs is more than one.
3.	It takes moderate amount of time.	It takes less time for job processing.
4.	In this, one by one job is being executed at a time.	In this, more than one process can be executed at a time.
5.	It is economical.	It is economical.
6.	The number of users is more than one.	The number of users is can be one or more than one.
7.	Throughput is moderate.	Throughput is maximum.
8.	Its efficiency is moderate.	Its efficiency is maximum.
9.	It is of two types: Single user multitasking and Multiple user multitasking.	It is of two types: Symmetric Multiprocessing and Asymmetric Multiprocessing.
10.	Number of user tasks is more than one.	Number of user tasks can be one or more than one.