**Operating Systems Questions**

**What is a Thread?**  
A thread is a path of execution within a process. A process can contain multiple threads.

**Why Multithreading?**  
A thread is also known as lightweight process. The idea is to achieve parallelism by dividing a process into multiple threads. For example, in a browser, multiple tabs can be different threads. MS Word uses multiple threads: one thread to format the text, another thread to process inputs, etc. More advantages of multithreading are discussed below

**Process vs Thread?**  
The primary difference is that threads within the same process run in a shared memory space, while processes run in separate memory spaces.  
Threads are not independent of one another like processes are, and as a result threads share with other threads their code section, data section, and OS resources (like open files and signals). But, like process, a thread has its own program counter (PC), register set, and stack space.

**Advantages:**

*1. Responsiveness:*If the process is divided into multiple threads, if one thread completes its execution, then its output can be immediately returned.

*2. Faster context switch:*Context switch time between threads is lower compared to process context switch. Process context switching requires more overhead from the CPU.

*3. Effective utilization of multiprocessor system:*If we have multiple threads in a single process, then we can schedule multiple threads on multiple processor. This will make process execution faster.

*4. Resource sharing:*Resources like code, data, and files can be shared among all threads within a process.  
Note: stack and registers can’t be shared among the threads. Each thread has its own stack and registers.

*5. Communication:*Communication between multiple threads is easier, as the threads shares common address space. while in process we have to follow some specific communication technique for communication between two process.

*6. Enhanced throughput of the system:*If a process is divided into multiple threads, and each thread function is considered as one job, then the number of jobs completed per unit of time is increased, thus increasing the throughput of the system.

What is thrashing?

Thrashing is a phenomenon in virtual memory scheme when the processor spends most of its time in swapping pages, rather than executing instructions.

### What are the four necessary and sufficient conditions behind the deadlock?

These are the 4 conditions:

1) **Mutual Exclusion Condition**: It specifies that the resources involved are non-sharable.

2) **Hold and Wait Condition**: It specifies that there must be a process that is holding a resource already allocated to it while waiting for additional resource that are currently being held by other processes.

3) **No-Preemptive Condition**: Resources cannot be taken away while they are being used by processes.

4) **Circular Wait Condition**: It is an explanation of the second condition. It specifies that the processes in the system form a circular list or a chain where each process in the chain is waiting for a resource held by next process in the chain

### What is deadlock? Explain.

Deadlock is a specific situation or condition where two processes are waiting for each other to complete so that they can start. But this situation causes hang for both of them

There are 4 necessary conditions to achieve a deadlock:

* **Mutual Exclusion:** At least one resource must be held in a non-sharable mode. If any other process requests this resource, then that process must wait for the resource to be released.
* **Hold and Wait:** A process must be simultaneously holding at least one resource and waiting for at least one resource that is currently being held by some other process.
* **No preemption:** Once a process is holding a resource ( i.e. once its request has been granted ), then that resource cannot be taken away from that process until the process voluntarily releases it.
* **Circular Wait:** A set of processes { P0, P1, P2, . . ., PN } must exist such that every P[ i ] is waiting for P[ ( i + 1 ) % ( N + 1 ) ].

### What is spooling?

Spooling is a process in which data is temporarily gathered to be used and executed by a device, program or the system. It is associated with printing. When different applications send output to the printer at the same time, spooling keeps these all jobs into a disk file and queues them accordingly to the printer.

### What are the states in the lifecycle of a Thread?

A thread can have one of the following states during its lifetime:

1. **New:** In this state, a Thread class object is created using a new operator, but the thread is not alive. Thread doesn't start until we call the start() method.
2. **Runnable:** In this state, the thread is ready to run after calling the start() method. However, the thread is not yet selected by the thread scheduler.
3. **Running:** In this state, the thread scheduler picks the thread from the ready state, and the thread is running.
4. **Waiting/Blocked:** In this state, a thread is not running but still alive, or it is waiting for the other thread to finish.
5. **Dead/Terminated:** A thread is in terminated or dead state when the run() method exits.

### What do you understand by inter-thread communication?

* The process of communication between synchronized threads is termed as inter-thread communication.
* Inter-thread communication is used to avoid thread polling in Java.
* The thread is paused running in its critical section, and another thread is allowed to enter (or lock) in the same critical section to be executed.
* It can be obtained by wait(), notify(), and notifyAll() methods.

### What is context switching?

In Context switching the state of the process (or thread) is stored so that it can be restored and execution can be resumed from the same point later. Context switching enables the multiple processes to share the same CPU

### How to detect a deadlock condition? How can it be avoided?

We can detect the deadlock condition by running the code on cmd and collecting the Thread Dump, and if any deadlock is present in the code, then a message will appear on cmd.

**Ways to avoid the deadlock condition in Java:**

* **Avoid Nested lock:** Nested lock is the common reason for deadlock as deadlock occurs when we provide locks to various threads so we should give one lock to only one thread at some particular time.
* **Avoid unnecessary locks:** we must avoid the locks which are not required.
* **Using thread join:** Thread join helps to wait for a thread until another thread doesn't finish its execution so we can avoid deadlock by maximum use of join method.

**Producer Consumer Problem:**

Consider the standard producer-consumer problem. Assume, we have a buffer of 4096 byte length. A producer thread collects the data and writes it to the buffer. A consumer thread processes the collected data from the buffer. Objective is, both the threads should not run at the same time.

**Using Mutex:**

A mutex provides mutual exclusion, either producer or consumer can have the key (mutex) and proceed with their work. As long as the buffer is filled by producer, the consumer needs to wait, and vice versa.

At any point of time, only one thread can work with the *entire* buffer. The concept can be generalized using semaphore.

**Using Semaphore:**

A semaphore is a generalized mutex. In lieu of single buffer, we can split the 4 KB buffer into four 1 KB buffers (identical resources). A semaphore can be associated with these four buffers. The consumer and producer can work on different buffers at the same time.

There is an ambiguity between *binary semaphore* and *mutex*. We might have come across that a mutex is binary semaphore. *But they are not*! The purpose of mutex and semaphore are different. May be, due to similarity in their implementation a mutex would be referred as binary semaphore.

Strictly speaking, a mutex is **locking mechanism** used to synchronize access to a resource. Only one task (can be a thread or process based on OS abstraction) can acquire the mutex. It means there is ownership associated with mutex, and only the owner can release the lock (mutex).

Semaphore is **signaling mechanism** (“I am done, you can carry on” kind of signal). For example, if you are listening songs (assume it as one task) on your mobile and at the same time your friend calls you, an interrupt is triggered upon which an interrupt service routine (ISR) signals the call processing task to wakeup.

**Causes of Thrashing :**

1. **High degree of multiprogramming**: If the number of processes keeps on increasing in the memory than number of frames allocated to each process will be decreased. So, less number of frames will be available to each process. Due to this, page fault will occur more frequently and more CPU time will be wasted in just swapping in and out of pages and the utilization will keep on decreasing.

For example:

Let free frames = 400

**Case 1**: Number of process = 100

Then, each process will get 4 frames.

**Case 2**: Number of process = 400

Each process will get 1 frame

Case 2 is a condition of thrashing, as the number of processes are increased, frames per process are decreased. Hence CPU time will be consumed in just swapping pages

1. **Lacks of Frames**:If a process has less number of frames then less pages of that process will be able to reside in memory and hence more frequent swapping in and out will be required. This may lead to thrashing. Hence sufficient amount of frames must be allocated to each process in order to prevent thrashing.

**Recovery of Thrashing :**

* Do not allow the system to go into thrashing by instructing the long term scheduler not to bring the processes into memory after the threshold.
* If the system is already in thrashing then instruct the mid term schedular to suspend some of the processes so that we can recover the system from thrashing.