Deadlock in Operating Systems and Java

1. Introduction

In a multiprogramming or multithreaded environment, several processes or threads may compete for limited resources. When each process holds one resource and waits indefinitely for another resource held by another process, a **deadlock** occurs.

Deadlock is a critical problem in **Operating Systems** and **Java multithreading**.

---

2. Definition of Deadlock

A **deadlock** is a situation where a set of processes or threads are blocked forever because each is waiting for a resource that is held by another process in the set.

> In deadlock, no process can proceed and the system comes to a halt for those processes.

---

3. Deadlock Example (Real-Life)

- Two people meet on a narrow bridge and each refuse to move back and both wait forever.

This situation represents a **deadlock**.

---

4. Coffman's Conditions for Deadlock

Deadlock can occur **if and only if all four conditions hold simultaneously**.

A. Mutual Exclusion

At least one resource must be held in a **non-shareable mode**

B. Hold and Wait

A process holds at least one resource and waits for additional resources

C. No Preemption

Resources cannot be forcibly taken from a process

D. Circular Wait

A circular chain of processes exists where each process waits for a resource held by the next

---

5. Deadlock in Java (Thread Example)

```
class DeadlockDemo
{
	public static void main(String[] args)
	{
		final String R1 = "Resource1";
		final String R2 = "Resource2";

		Thread t1 = new Thread(() ->
		{
			synchronized (R1)
			{
				System.out.println("Thread 1: Locked R1");
				try { Thread.sleep(100); } catch (Exception e) {}
				synchronized (R2)
				{
					System.out.println("Thread 1: Locked R2");
				}
			}
		});

		Thread t2 = new Thread(() ->
		{
			synchronized (R2)
			{
				System.out.println("Thread 2: Locked R2");
				try { Thread.sleep(100); } catch (Exception e) {}
				synchronized (R1)
				{
					System.out.println("Thread 2: Locked R1");
				}
			}
		});

		t1.start();
		t2.start();
	}
}
```

Conclusion

Deadlock is a serious synchronization problem in concurrent systems. Understanding its conditions and handling techniques helps in designing efficient and safe multithreaded and operating systems.