Operating Systems
INF333

TP07
Deadlocks

Eda Bahar 09/05/2025
ebahar@gsu.edu.tr
edaabahar@gmail.com

About this TP

- In this TP, you will be learning:
  - Deadlocks

The Deadlock Problem

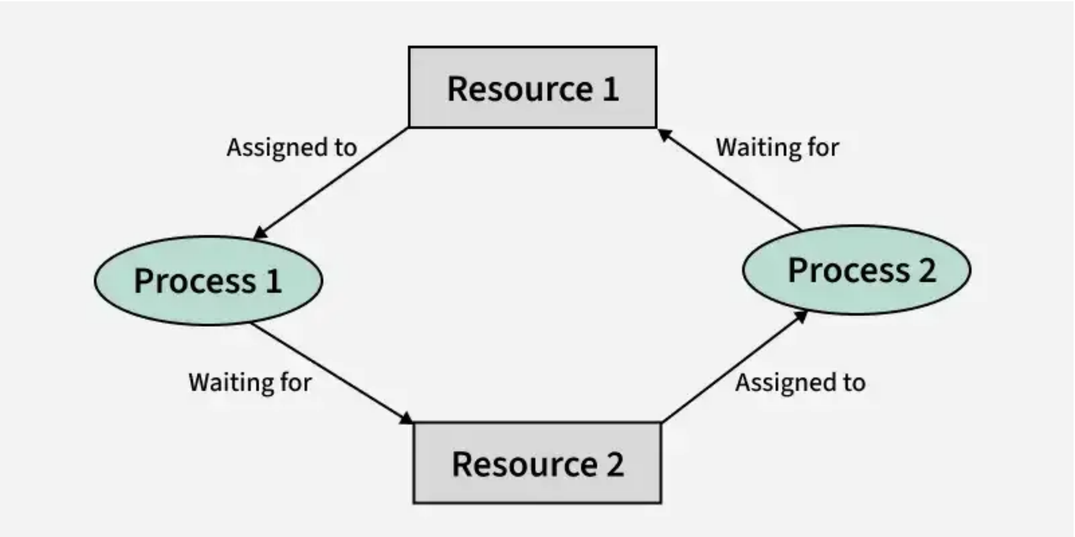
Definition

- **Deadlock** is a situation in computing where two or more processes are unable to proceed because each is waiting for the other to release resources.
- Where:
  - Mutual Exclusion (Mutex)
  - Resource Holding
  - Circular Wait
  - No preemption

How Does Deadlock occur in the Operating System?

- A process in an operating system uses resources in the following way:

  1. Requests a resource
  2. Uses the resource
  3. Releases the resource



Deadlock Examples

- Semaphores A and B are initialized to 1.
  P0 executes wait(A) and preempts
  P1 executes wait(B)
  Now P0 and P1 enter in deadlock!

P0	P1
wait(A);	wait(B);
wait(B);	wait(A);

- Two threads, each locking two resources, but in opposite order.
  Thread 1 locks A, then Thread 2 locks B
  Then each tries to lock what the other is holding -> deadlock!

```
// Thread 1
lock(mutexA);
sleep(1);
lock(mutexB);

// Thread 2
lock(mutexB);
sleep(1);
lock(mutexA);
```

Deadlock Examples

- Process A locks file1 and wants file2.
  Process B locks file2 and wants file1. -> deadlock!

- Process A waits to **receive** a message from Process B
  Process B waits to **receive** a message from Process A
  Since both are waiting to receive, neither sends → deadlock!

Deadlock Examples

Dining Philosopher Problem

- The Dining Philosopher Problem states that K philosophers are seated around a circular table with one chopstick between each pair of philosophers.
- 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.
- In order to avoid deadlock or starvation, a solution must be implemented that ensures that each philosopher can access the resources they need to perform their task without interference from other philosophers.