كلية الحاسبات والذكاء الإصطناعي
Faculty of Computers & Artificial Intelligence

جامعة حلوان
HELWAN UNIVERSITY

Cybersecurity Diploma
Operating System

Project 3
Case # (3) - Dining Philosopher's problem

Realistic Scenario (Shared Clipboard Access)

prepared by

Ahmed Mohamed Maged
Wesam Ahmed Hassan
Mohamed Tarek Taha Ahmed
Mahmoud Ahmed AbdelAziz
Mahmoud Osama Ibrahim

Supervised by

Dr. Ahmed Hesham

Documentation

1. Introduction:

Imagine 5 programs (simulated as threads) trying to read from and write to a shared clipboard. Only one program can access the clipboard at a time to ensure consistency. The clipboard content changes based on what each program writes to it.

---

2. Pseudocode:

```
Start
    Initialize Clipboard with initial content
    Create a Lock to ensure sequential access to the shared resource

    For each Thread (Program):
        - Randomly choose an action: read or write
        - Acquire the lock to ensure no other thread accesses the resource
simultaneously
        - If the action is read, read the content from the clipboard
        - If the action is write, update the clipboard with new content
    End for each thread

    Wait for all threads to complete
End
```

3. Solution Explanation:

- **We use multithreading where each program (thread) runs concurrently. Each thread either tries to read from or write to the shared clipboard**.
- **A Lock is used to ensure that access to the clipboard is sequential, meaning only one thread can access it at any given time.**
- **Random actions (reading or writing) simulate real-world scenarios where users might perform tasks on a shared resource.**

4. Deadlock Example and Solution

- **Deadlock Scenario:**
  - **Deadlock can occur when two or more threads hold a lock and each one waits for the other to release a lock, resulting in a cycle where none of the threads can proceed, If there were two locks and threads acquired one lock each and waited for the other, a deadlock would occur.**

- **Solution:**
  - **This solution uses a single lock, so no circular dependencies can form, preventing deadlock.**

- **Deadlock Prevention:**
  - **There is no risk of Deadlock in this solution because only one lock is used per thread, and threads don’t wait on each other in a circular manner.**
  - **Each thread simply locks the resource when it needs access and releases the lock afterward.**

---

5. Starvation Example and Solution

- **Starvation Scenario:**
  - **happens when one thread is unable to gain access to a shared resource because other threads are consistently allowed to access it, If threads with certain priorities were always granted access, other threads might never get a chance to use the resource**

- **Solution:**
  - **This solution randomly chooses between reading and writing actions, giving each thread a fair chance to access the clipboard. Additionally, no thread is prioritized over others, which helps avoid starvation.**

- **Starvation Prevention:**
  - **The program randomly selects whether to read or write, which ensures that no thread is indefinitely "starved" of the chance to access the clipboard.**
  - **Since the operations are random, each thread has a fair opportunity to access the clipboard.**