# CS 3502: Multi-Threaded Programming and IPC Project

Aly Traore CS 3502/W02 NETID: Atraor13@students.kennesaw.edu

March 2025

## 1 Introduction

This project demonstrates multi-threading and inter-process communication (IPC) in C. The goals are to create a multi-threaded application using true parallel threads, apply synchronization mechanisms such as mutexes, and implement IPC using named pipes.

## 2 Implementation Details

### 2.1 Multi-Threading Implementation

The project follows a four-phase threading approach:

- **Phase 1: Basic Multi-Threading** - Creates multiple threads to execute concurrent operations.
- **Phase 2: Synchronization with Mutex** - Implements mutexes to prevent race conditions.
- **Phase 3: Deadlock Demonstration** - Simulates a deadlock scenario using multiple mutexes.
- **Phase 4: Deadlock Resolution** - Implements a proper resource acquisition strategy to prevent deadlocks.

### 2.2 IPC Implementation

IPC is implemented using Named Pipes to allow communication between separate processes:

- A **Pipe Server** listens for connections and sends messages.
- A **Pipe Client** connects to the server and receives messages.
- The server is extended to handle multiple clients concurrently.

# 3 Challenges and Solutions

- **Thread Synchronization** - Using mutexes prevented race conditions but required careful resource management.
- **Deadlocks** - Implementing proper locking sequences avoided circular wait conditions.
- **IPC Handling** - Named Pipes needed proper connection handling for multiple clients.

# 4 Results and Testing

## 4.1 Threading Testing

- Verified thread execution using console logs.
- Tested mutex locks by running simultaneous operations on shared resources.
- Simulated deadlock and confirmed process freezing.
- Applied deadlock resolution and ensured smooth execution.

## 4.2 IPC Testing

- Verified client-server communication using Named Pipes.
- Ran multiple clients to test concurrent connections.
- Ensured proper data transmission and error handling.

# 5 Conclusion

This project enhanced understanding of multi-threading and IPC in C. Implementing synchronization mechanisms, handling deadlocks, and using Named Pipes for process communication provided valuable experience in concurrent programming.