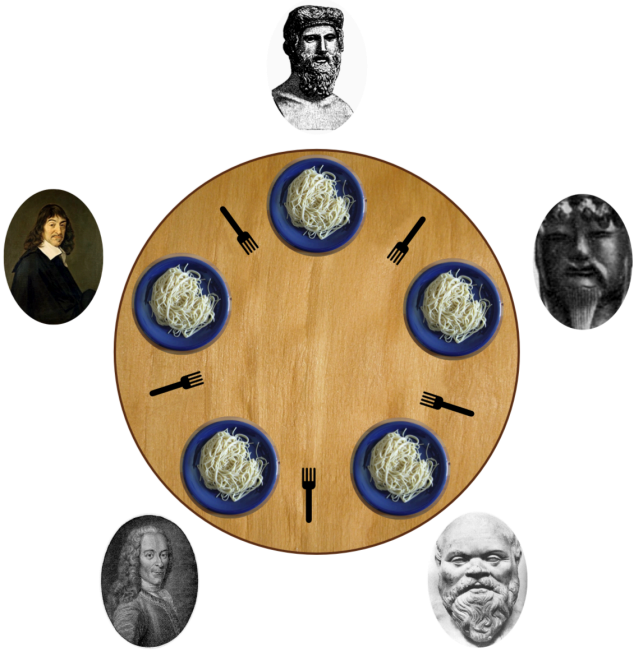
Group Project 2

Synchronization

03/15/2024

CECS 326 - Operating Systems



Brandon Quach (#029133232)

Michael Kim (#025633788)

Project demo:

Link: https://youtu.be/yPLujjVHudw

Project Description:

In this project, we revisited the dining- philosophers problem based on our presentation slides. The project involves executing a solution to this issue utilizing either POSIX mutex locks and condition variables or Java condition variables. Our team decided to use Java for this project. The program will create five philosophers, each identified by numbers 0 through 4, with each philosopher functioning as an independent thread. These philosophers will oscillate between contemplating and dining. To mimic the action, each philosopher will sleep for a random duration ranging from 1 to 3 seconds.

Steps:

1. Download the diningPhilosophers.java, diningServer.java, diningServerImpl.java, and philsopher.java
2. Navigate to the folder using cd where your files are saved and compile them using these commands
    a. javac diningPhilosophers.java
    b. javac diningServer.java
    c. javac diningServerImpl.java
    d. javac philosopher.java
3. Run the main file using the command “java diningPhilosophers.java”
4. Since the program will be going forever in terminal, use CTRL + c to stop the program

Code analysis:

diningServer.java contains methods called by the philosophers like takeForks and returnForks.

```
public interface DiningServer
{
    // Called by a Philosopher when they wish to eat
    public void takeForks(int philosopher_Num);

    // Called by a Philosopher when they are finished eating
    public void returnForks(int philosopher_Num);
}
```

diningServerImpl.java also contains methods takeForks and returnForks but has the implementation.

```
@Override
// Method to handle philosophers picking up forks to eat
// @param philNumber: unique indentifier of the philosopher
public void takeForks(int philosopherID) {
    int leftFork = philosopherID;
    int rightFork = (philosopherID + 1) % forks.length;

    // Lock the left fork
    forks[leftFork].lock();
    // Lock the right fork
    forks[rightFork].lock();
}
```

```
    @Override
    // Method to handle the Philosophers when they put down the forks after eating to think
    // @param is the unique identifier of the philosopher
    public void returnForks(int philosopherID) {
        int leftFork = philosopherID;
        int rightFork = (philosopherID + 1) % forks.length;

        // Unlock the left fork
        forks[leftFork].unlock();
        // Unlock the right fork
        forks[rightFork].unlock();
    }
}
```

philosopher.java contains the philosopher threads that alternate between eating and thinking. This class is where each philosopher is identified by a number 0 - 4 and the philosophers simulate eating, thinning, and sleeping for a random period between 1 - 3 seconds.

```java
@Override
public void run() {
    while (true) {
        // Randomly choose between 1, 2, or 3 seconds (1000, 2000, or 3000 milliseconds) for thinking and eating
        int thinkingDuration = (int) (Math.random() * 3) + 1;
        int eatingDuration = (int) (Math.random() * 3) + 1;
        long thinkingTimeMillis = thinkingDuration * 1000;
        long eatingTimeMillis = eatingDuration * 1000;

        // Record the start time before the philosopher starts thinking
        long startTime = System.currentTimeMillis();

        // Indicate that the philosopher is thinking
        System.out.println("Philosopher " + philosopherNum + " is thinking for " + thinkingDuration + " seconds.");

        // Simulate thinking for random chosen duration
        try {
            Thread.sleep(thinkingTimeMillis);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        // Record the start time before the philosopher tries to pick up forks
        startTime = System.currentTimeMillis();

        //Attempt to acquire forks
        diningSer.takeForks(philosopherNum);

        // Print a message indicating which philosopher has the forks
        System.out.println("Philosopher " + philosopherNum + " has the forks and is eating for " + eatingDuration + " seconds");
```

```java
// Simulate eating for the randomly chosen duration
try {
    Thread.sleep(eatingTimeMillis);
} catch (InterruptedException e) {
    e.printStackTrace();
}

// Here the Philosophers put down forks and think
diningSer.returnForks(philosopherNum);
```

Program output:

```
Philosopher 4 is thinking for 2 seconds.
Philosopher 3 has the forks and is eating for 1 seconds
Philosopher 3 is thinking for 1 seconds.
Philosopher 2 has the forks and is eating for 3 seconds
Philosopher 2 is thinking for 2 seconds.
Philosopher 1 has the forks and is eating for 2 seconds
Philosopher 1 is thinking for 3 seconds.
Philosopher 0 has the forks and is eating for 2 seconds
Philosopher 4 has the forks and is eating for 1 seconds
Philosopher 0 is thinking for 3 seconds.
Philosopher 4 is thinking for 2 seconds.
Philosopher 3 has the forks and is eating for 2 seconds
Philosopher 3 is thinking for 2 seconds.
Philosopher 2 has the forks and is eating for 2 seconds
Philosopher 2 is thinking for 3 seconds.
Philosopher 1 has the forks and is eating for 1 seconds
Philosopher 1 is thinking for 2 seconds.
Philosopher 0 has the forks and is eating for 1 seconds
Philosopher 0 is thinking for 2 seconds.
Philosopher 4 has the forks and is eating for 3 seconds
```

What each individual did:

For this project we kept the roles the same to avoid any confusion.

Brandon: Brandon took the role in the completion of the lab report and ensuring that all project deliverables were included. In addition to his lab report responsibilities, he also worked on the code with Michael and input the proper comments to show the design.

Michael: Michael took a technical role within the group, with a primary focus on both software. His responsibilities encompassed software development, creation, and software components. In addition to his software responsibilities, he also focused on the video.

Work split was mostly 50/50 when it came to the software and its deliverables.

References:

https://www.youtube.com/watch?v=JEJdQN0mcG4

https://www.baeldung.com/java-dining-philoshophers