Course : Operating Systems

Assignment 02

Report: Reading sensor's water level with Semaphores

Members	ID
Quentin Lauriot	40304712
Panying Song	40304754
Sarah Nguyen	40304753

Concordia University & Efrei

Table of content

I. Introduction

- ## Choice

Just like the first assignment, we decided to code in C++ instead of Java to challenge ourselves more and learn how to use this language.

- ## Problem statement

A water company has designed a water tank system to store water for their customer. The system used the sensor to measure the water level inside the tank. The hardware is already assembled and tested for you by producing data.

Your task is to implement the software that reads the sensor's water level. If the tank is empty, the application should signal the water faucet to start producing water. Then, if the tank is full, the water faucet must stop to eliminate wasting water. As a result, your system is Intelligent designed to provide water at a critical point and save water and power consumption.

II. Code Description

- ## Description

Our program defines a WaterSensor class that simulates a water tank with functionality to produce and consume water based on specified thresholds and conditions. It automatically triggers water production or consumption based on the current water level compared to the criticalPoint. This design helps to optimize water usage, avoiding overflows or shortages in the tank.

```
class WaterSensor {
private:
    double waterLevel;
    double criticalPoint;
    std::binary_semaphore provideWater;
    std::binary_semaphore consumeWater;

public:
    WaterSensor(double criticalPoint);
    WaterSensor(double criticalPoint, double initialWaterLevel);

    void produce();
    void consume();
};
```

- ## Private Members
  - waterLevel: Current water level in the tank.
  - criticalPoint: Water level threshold indicating when the tank is critically low.
  - provideWater: A binary semaphore to control water production.
  - consumeWater: A binary semaphore to control water consumption.

- ## Functionnality & Actions

  - **Water Production**

The produce() function simulates water production adding an amount of water to produce.

It starts by acquiring provideWater semaphore to begin production. If the waterLevel falls below a critical point (criticalPoint), the produce() method is triggered. Once the waterLevel reaches or exceeds the criticalPoint, it signals the consumeWater semaphore to indicate that water is available for consumption.

```
// Simulate water production process
void WaterSensor::produce()
{
    std::random_device rd;
    while (1) {
        provideWater.acquire(); // Wait until allowed to produce water

        // Simulate random water production
        std::mt19937 gen(rd());
        std::uniform_int_distribution<> dis(5, MAX_LEVEL / 4);
        double amountProduced = dis(gen);
        for (int i = 0; i < amountProduced; i++) {
            if (waterLevel < MAX_LEVEL) {
                waterLevel++;
                std::cout << BLUE << "(Producing water): Tank capacity = " + std::to_string(waterLevel) + "%\n";
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
        }

        // Check if water level is below or at the critical point
        if (criticalPoint >= waterLevel) {
            std::cout << RED << "Tank is equal or lower than critical point\n" << RESET;
            provideWater.release();  // Allow more water production
        }
        else {
            consumeWater.release(); // Signal that water is available for consumption
        }

        std::this_thread::sleep_for(std::chrono::seconds(1));
    }
}
```

- **Water Consumption**

This function simulates water consumption by decrementing an amount of water to consume.

It repeatedly acquires the consumeWater semaphore to start consumption. If the waterLevel exceeds the criticalPoint, the consume() method is triggered. If the waterLevel drops below criticalPoint, it signals the provideWater semaphore to allow more water production.

```
// Simulate water consumption process
void WaterSensor::consume()
{
    std::random_device rd;
    while (1) {
        consumeWater.acquire(); // Wait until allowed to consume water

        std::mt19937 gen(rd());
        std::uniform_int_distribution<> dis(1, waterLevel - 1);
        double amountConsumed = dis(gen);

        // Check if water level is above critical point
        if (waterLevel > criticalPoint) {
            for (int i = 0; i < amountConsumed; i++) {
                waterLevel--;
                std::cout << MAGENTA << "(Consuming water): Tank capacity = " + std::to_string(waterLevel) + "%\n";
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
        }

        // Check if water level is below maximum capacity
        if (waterLevel < MAX_LEVEL) {
            provideWater.release(); // Allow more water production
        }
        else {
            consumeWater.release(); // Signal that water is available for consumption
        }
        std::this_thread::sleep_for(std::chrono::seconds(1));
    }
}
```

- **Role of Threads and Semaphores**

In our code, threads producerThread and consumerThread simulate concurrent processes of water production and consumption to allow actions to occur simultaneously. For provideWater and consumeWater which are semaphores, they control access to the shared resource waterLevel to ensure proper sequencing and coordination between both methods. Thus, semaphores manage the availability of water and coordinate actions between threads to prevent race conditions (overproduction or overconsumption).

```
int main() {
    WaterSensor sensor(15);
    std::thread producer_thread(std::bind(&WaterSensor::produce, &sensor));
    std::thread consumer_thread(std::bind(&WaterSensor::consume, &sensor));

    producer_thread.join();
    consumer_thread.join();

    return 0;
}
```

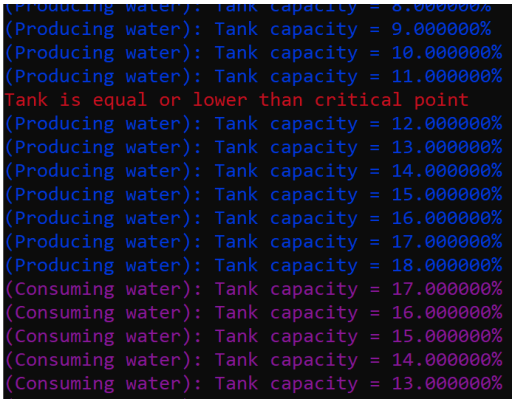
III. Demonstration

- **Initiating Water Production**

The critical point is set at 15% for this example. Initially, the tank is empty, so it begins to produce water continuously as long as the water level is below the critical point (15%).

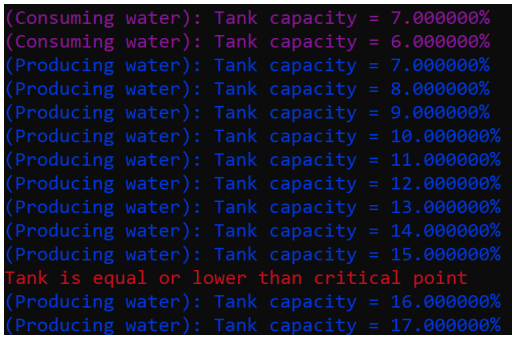
- **Starting Water Consumption**

Once the water level in the tank exceeds the critical point (15%), it can begin to consume some water.



- **Resuming Water Production**

If the water level drops below the critical point again, the system will resume water production without consuming it, as long as the level remains below the critical point.



- **Stopping Water Production**

When the tank reaches full capacity, the water faucet stops to prevent wasting water.