📦 Distributed Log-Based Message Queue (Mini Kafka)

Overview

This project is a **distributed log-based message queue** inspired by **Apache Kafka**, implemented from scratch in **C**.
It demonstrates how core messaging systems work internally, including **append-only logs, partitioning, offsets, consumer groups, persistence, and concurrency**.

Architecture

The system consists of three main components:

- **Producer** – Sends messages (orders) to the broker
- **Broker** – Stores messages durably and serves consumers
- **Consumer** – Reads messages sequentially using offsets

All communication happens over **TCP sockets**.

Key Concepts Implemented

1. Append-Only Log

- Each partition is an **append-only file on disk**
- Messages are never updated or deleted
- Guarantees **ordering** within a partition

2. Partitioning

- Topics are divided into multiple **partitions**
- Partition selection is **key-based**:

```
partition = key % number_of_partitions
```

- Ensures:
  - Ordering per key
  - Parallelism across partitions

3. Consumer Groups

- Consumers join a **consumer group**
- Each partition is assigned to **exactly one consumer per group**
- Enables **horizontal scaling** without duplication

4. Persistent Offsets

- Offsets are stored on disk per:

```
consumer-group + partition
```

- Consumers resume from the last committed offset after restart
- Matches Kafka's offset semantics

5. Indexing

- Each partition maintains an **index file**
- Index maps:

```
offset → file position
```

- Enables fast random access during fetch requests

6. Concurrency & Locking

- Broker uses **thread-per-connection**
- **Partition-level locks** protect writes
- Reads are **lock-free** for high throughput

7. Durability

- Messages are flushed using **fsync**
- Acknowledgement is sent only after data is safely written to disk

8. Network Protocol

- Custom **length-prefixed binary protocol**
- Prevents partial reads and corruption

- One request per TCP connection (simple & reliable)

Directory Structure

```
Distributed Log Storage/
├── Broker/
│   └── Broker.c
├── Producer/
│   └── Producer.c
├── Consumer/
│   └── Consumer.c
├── Data/        # Runtime logs (ignored in git)
│   └── partition-*/
├── offsets/     # Consumer offsets (ignored in git)
└── .gitignore
```

How to Run

Compile

```
gcc Broker/Broker.c -o Broker/Broker -lpthread gcc Producer/Producer.c -o
Producer/Producer gcc Consumer/Consumer.c -o Consumer/Consumer
```

Start Broker

```
./Broker/Broker
```

Start Consumers

```
./Consumer/Consumer A ./Consumer/Consumer B ./Consumer/Consumer C
```

Run Producer

```
./Producer/Producer
```

Guarantees Provided

- ✅ Message ordering per key
- ✅ At-least-once delivery
- ✅ Durable storage
- ✅ Parallel consumption
- ✅ Offset-based replay
- ✅ Fault tolerance across restarts