IPCManager

Overview

The **IPCManager** is a thread-safe Inter-Process Communication (IPC) module implemented in C++17. It provides a simple and efficient way to manage communication between threads or processes using a queue-based approach. The module is designed following **SOLID** principles, ensuring modularity, extensibility, and maintainability.

Features

- **Thread-safe queue**: Uses `std::queue` with `std::mutex` and `std::condition_variable` for synchronization.
- **Generic design**: Template-based implementation supports any data type (e.g., `std::vector<float>` for audio buffers).
- **High performance**: Tested to achieve >20 MB/s throughput with no data loss.
- **Comprehensive tests**: Unit tests using Google Test validate data integrity, throughput, and edge cases.

Prerequisites

- **Compiler**: C++17 compatible (e.g., g++ 9+)
- **Dependencies**:
  - CMake 3.10+
  - Google Test (libgtest-dev)
- **Optional**: Docker (for reproducible builds and tests)

Installation

Using Docker (Recommended)

1. Clone the repository:

   ```
   git clone <repository-url>
   cd ipcmanager
   ```

2. Build and run the Docker container:

   ```
   docker build -t ipcmanager-test .
   docker run --rm ipcmanager-test
   ```

   This will compile the code, run the unit tests, and display the results.

Manual Installation

1. Install dependencies on Ubuntu:

```
sudo apt-get update
sudo apt-get install -y g++ cmake libgtest-dev
```

2. Clone the repository:

```
git clone <repository-url>
cd ipcmanager
```

3. Build the project:

```
mkdir build
cd build
cmake ..
cmake --build . --config Release
```

4. Run the tests:

```
./IPCTest
```

Usage

The `IPCManager` module provides a simple API for sending and receiving data between threads or processes.

Example

```
#include "IPCManager.h"
#include <vector>
#include <thread>
#include <iostream>

int main() {
    IPCManager<std::vector<float>> ipc;
    std::vector<float> buffer(960, 1.0f); // 960-frame audio buffer

    // Producer thread
    std::thread producer([&]() {
        for (int i = 0; i < 10; ++i) {
```

```
            ipc.push(buffer);
            std::cout << "Sent buffer " << i << "\n";
        }
    });

    // Consumer thread
    std::thread consumer([&]() {
        for (int i = 0; i < 10; ++i) {
            std::vector<float> received;
            if (ipc.pop_blocking(received)) {
                std::cout << "Received buffer " << i << " with " <<
received.size() << " frames\n";
            }
        }
    });

    producer.join();
    consumer.join();
    return 0;
}
```

API

- `void push(const T& item)`: Adds an item to the queue.
- `std::optional<T> pop()`: Retrieves an item (non-blocking, returns `std::nullopt` if empty).
- `bool pop_blocking(T& item)`: Retrieves an item (blocking until data is available).

Project Structure

```
ipcmanager/
├── IPCManager.h          # Header file with IPCManager implementation
├── IPCManagerTest.cpp    # Unit tests using Google Test
├── Dockerfile            # Docker configuration for build and test
├── CMakeLists.txt        # CMake configuration
├── README.md             # This file
```

Testing

The project includes unit tests to validate:

- **Data Integrity**: Ensures no data loss when sending/receiving buffers.
- **Throughput**: Verifies >20 MB/s transfer rate for 960-frame audio buffers (48 kHz, mono).
- **Edge Cases**: Tests behavior with empty queues.

To run tests:

```
```

```
cd build
./IPCTest
```

Test Results

- **Data Integrity**: 1000 buffers (960 frames each) sent and received without loss.
- **Throughput**: >20 MB/s in multi-threaded producer-consumer scenarios.
- **Edge Cases**: Correct handling of empty queues.

Contributing

Contributions are welcome! Please:

1. Fork the repository.
2. Create a feature branch (`git checkout -b feature-name`).
3. Commit your changes (`git commit -m "Add feature"`).
4. Push to the branch (`git push origin feature-name`).
5. Open a Pull Request.

License

This project is licensed under the MIT License. See the `LICENSE` file for details.

Contact

For questions or feedback, please open an issue on the repository.