Design and implement a robust, distributed system using C++ that effectively leverages signals, sockets, and inter-process communication (IPC) to manage and coordinate multiple processes for a real-time data processing pipeline.

System Requirements

Data Ingestion: Continuously receive data from multiple sources (e.g., network sockets, files, sensors) and distribute it across multiple worker processes.

Data Processing: Distribute incoming data to multiple worker processes, each responsible for specific data transformations or calculations.

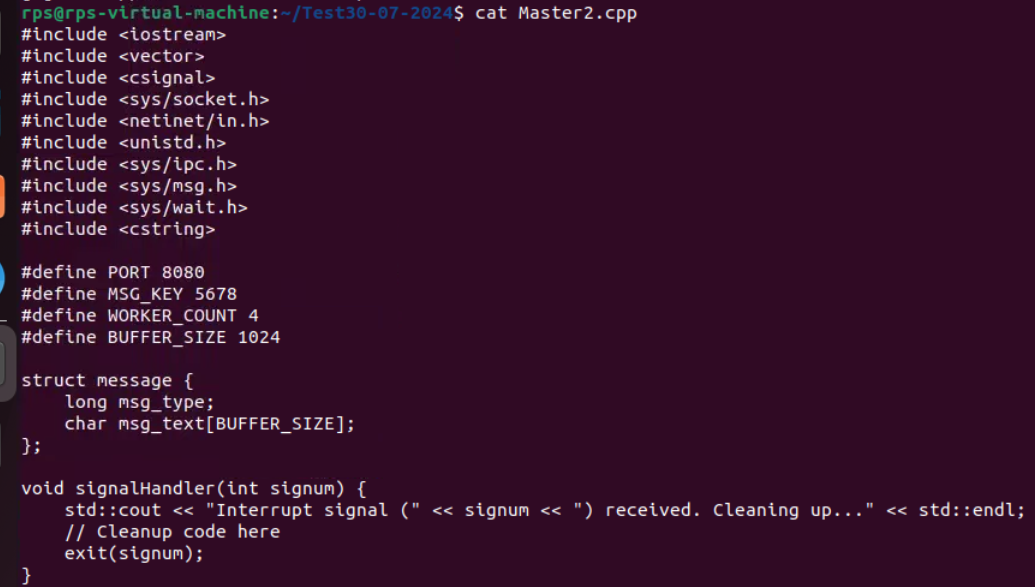
Error Handling: Implement robust error handling mechanisms using signals to gracefully handle unexpected events (e.g., process termination, network failures).

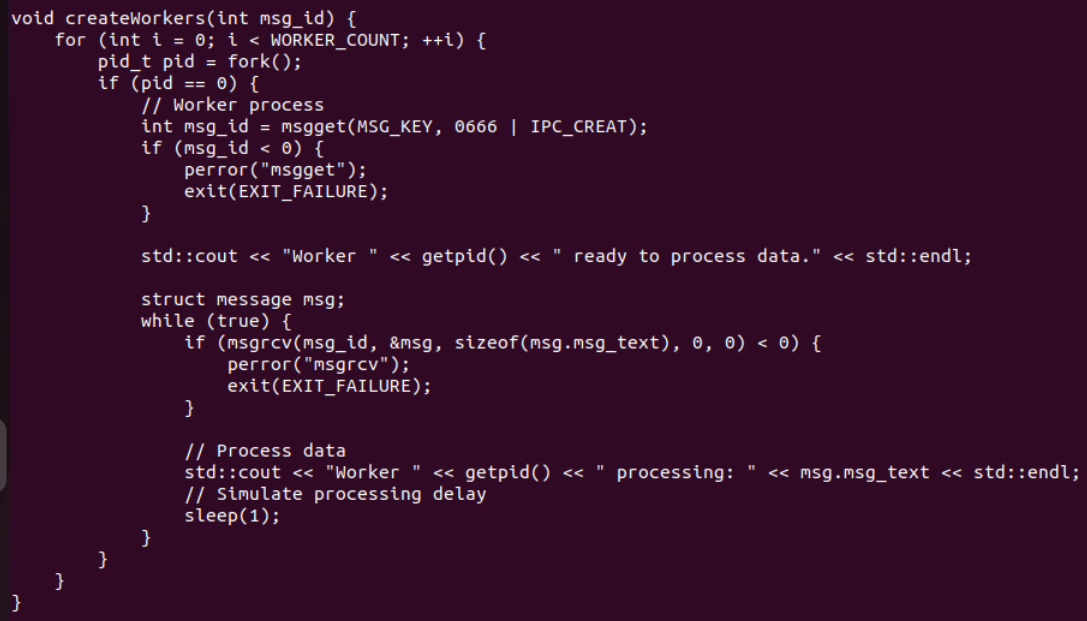
Inter-Process Communication: Utilize IPC (e.g., shared memory, message queues) for efficient communication and synchronization between processes.

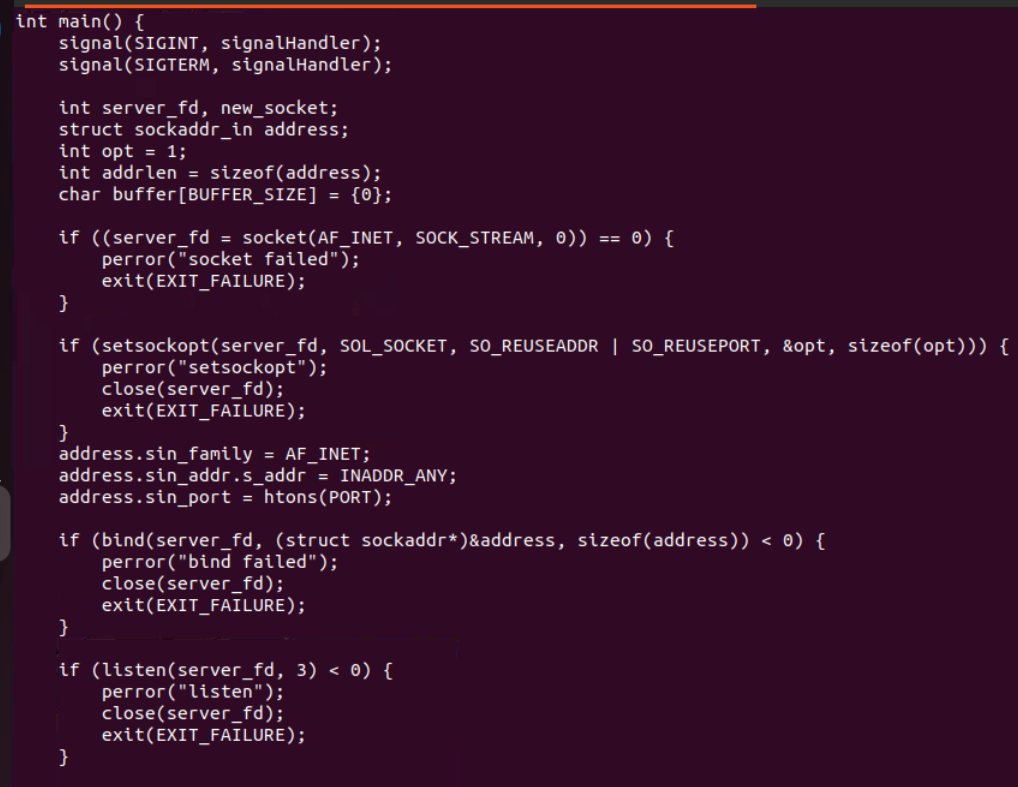
Performance Optimization: Optimize the system for low latency and high throughput, considering factors like network congestion, process scheduling, and data transfer efficiency.

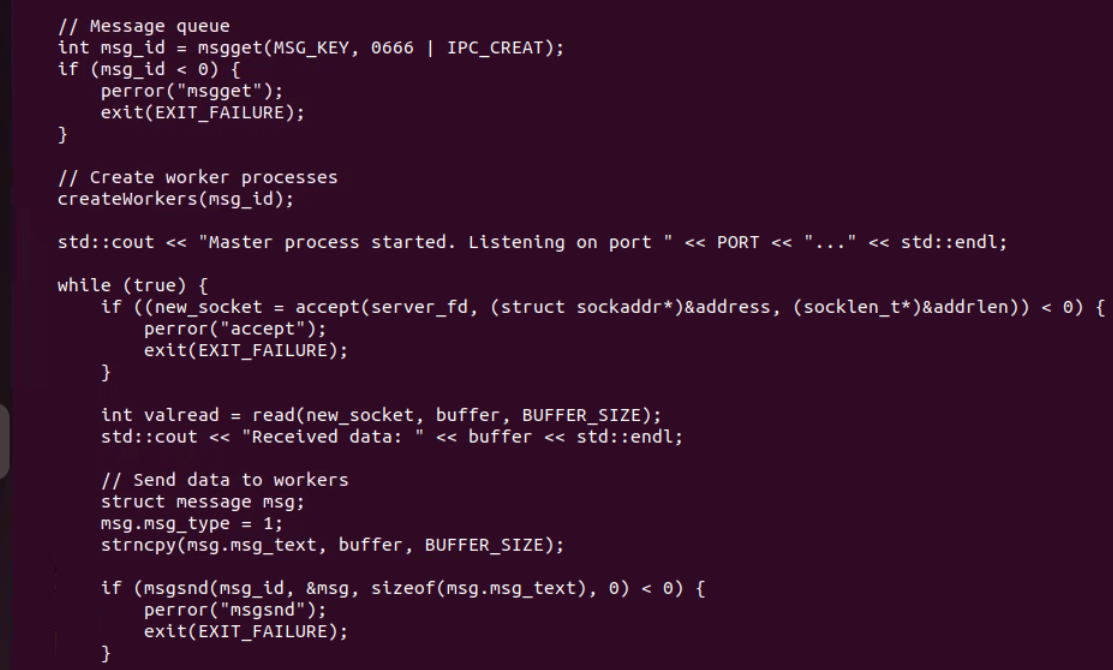
Scalability: Design the system to handle increasing data volumes and processing load by dynamically adjusting the number of worker processes.

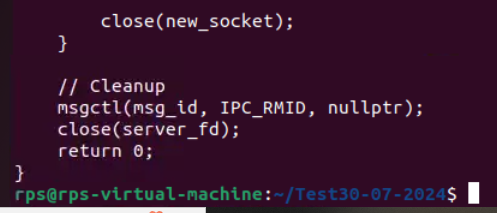
**Server CODE :**

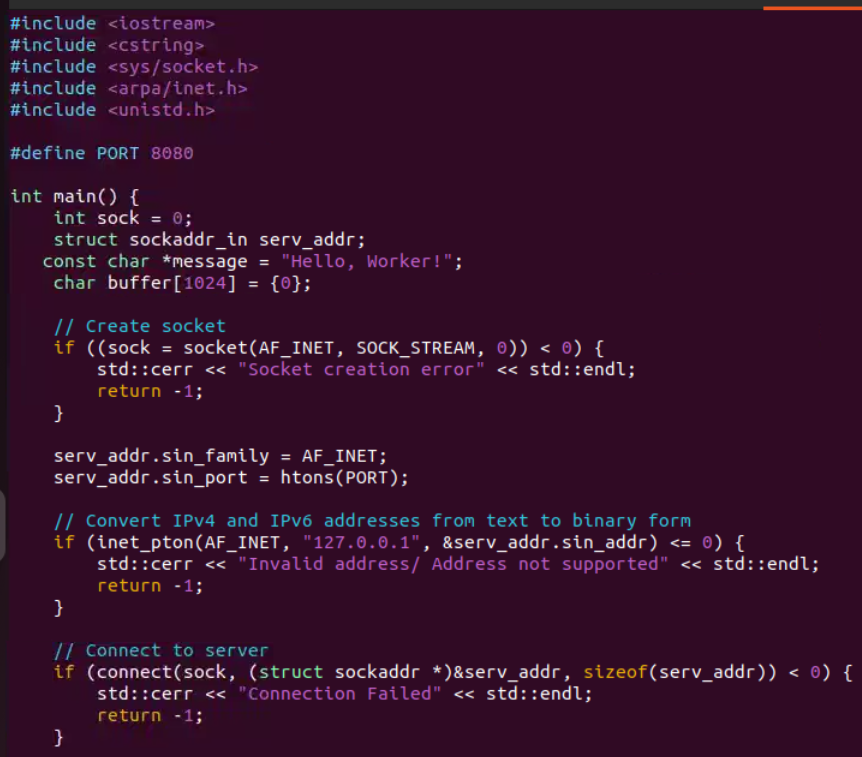
****

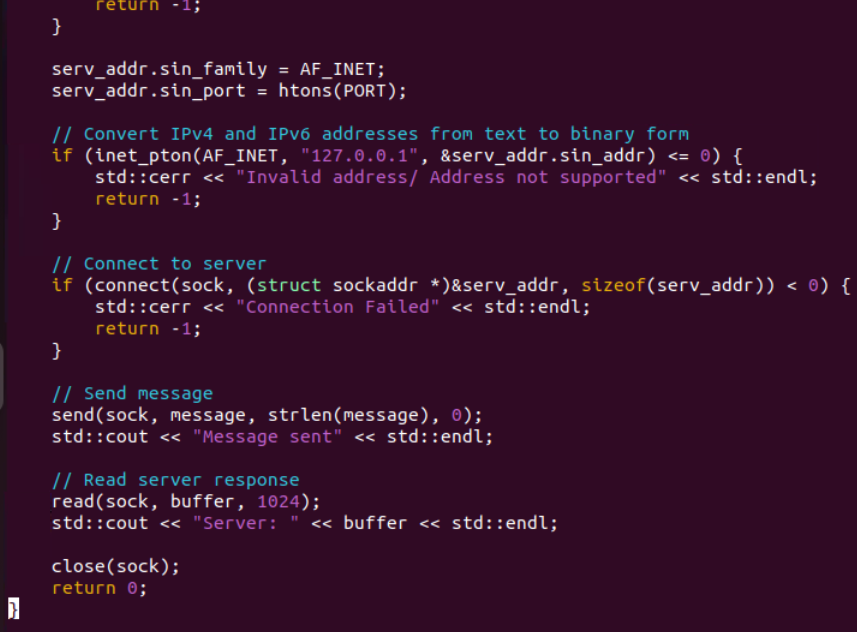
****

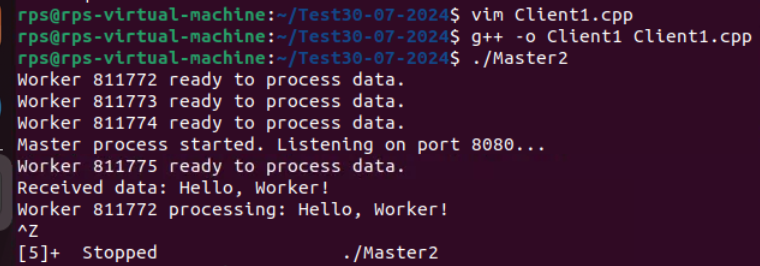
****

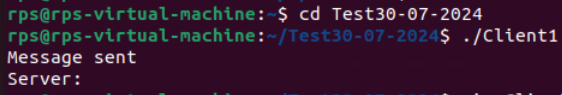
****

****

**CLIENT CODE:  
**

****

**OUTPUT:  
**

****

**TASK2:**

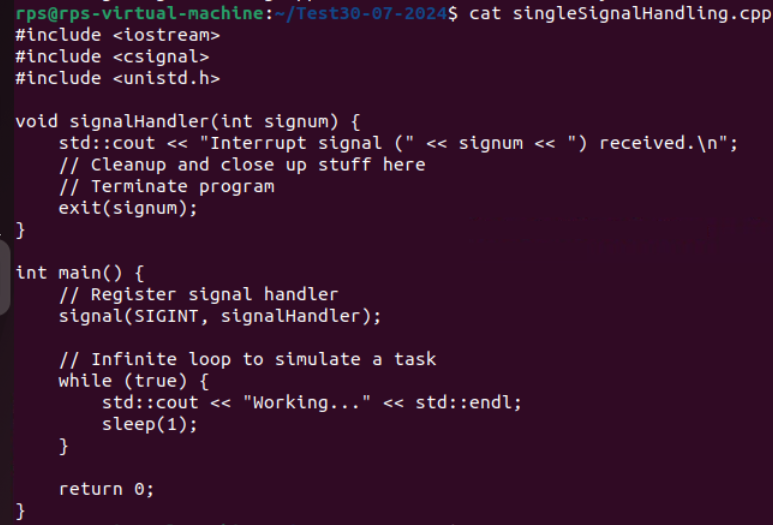
**Signal Handling:**

**Write a C++ program that sets up a signal handler for SIGINT. The program should perform some tasks and print a message when SIGINT is caught, then terminate gracefully.**

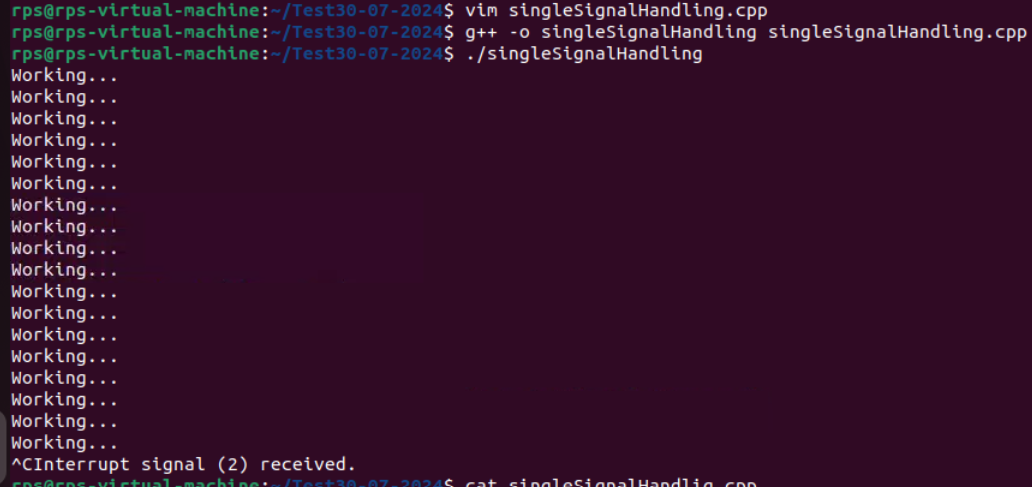
**How would you modify your program to handle multiple different signals, each with a unique handling function?**

**Single Signal Handling:**

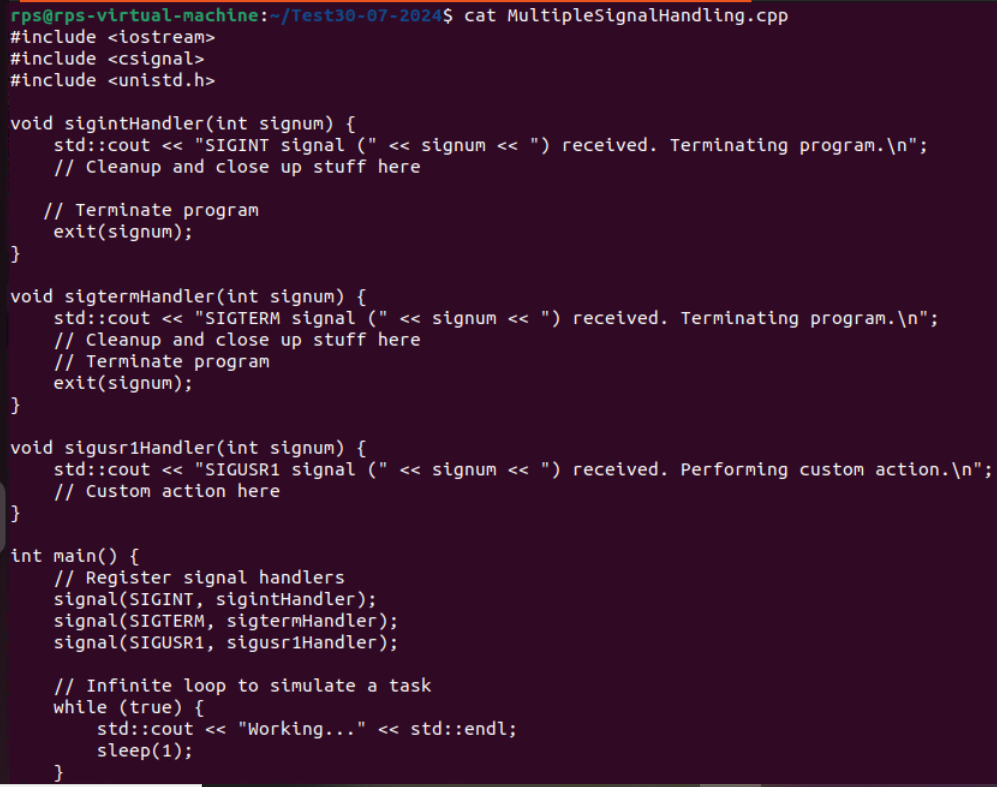
**CODE:**

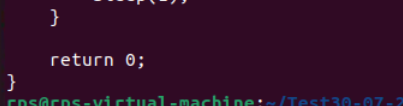
****

**Output:**

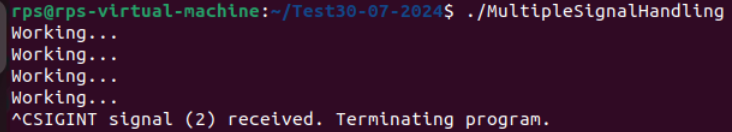
****

**MULTIPLE SIGNAL HANDLING:**

****

****

**OUTPUT:**

****

**TASK 3:**