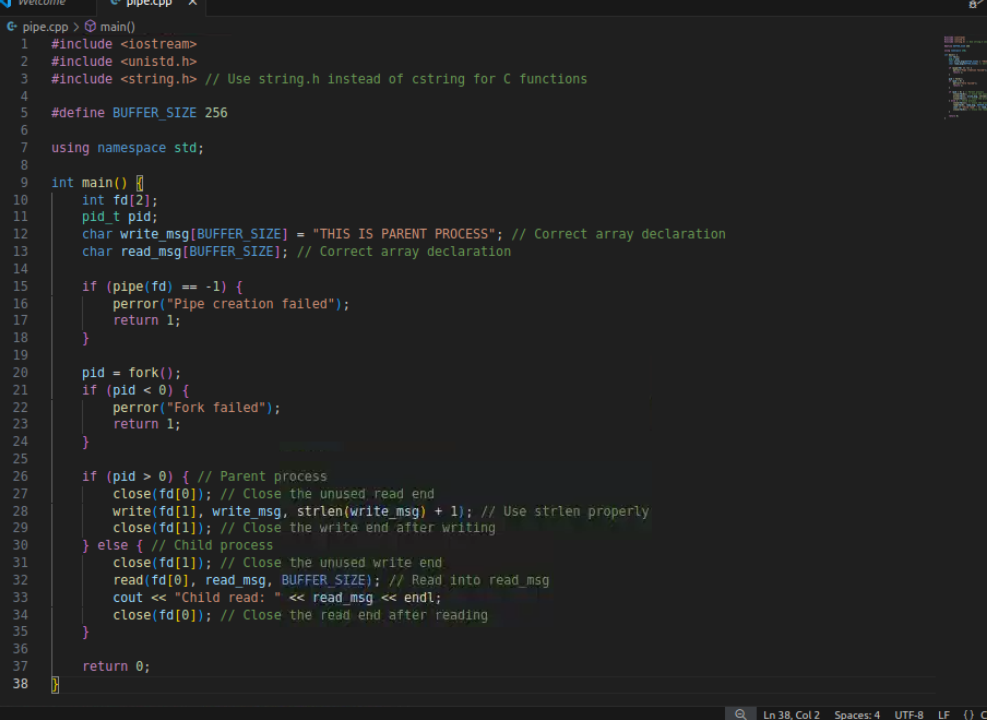
**DAY11 LSP ASSESSMENT**

**Problem Statement: Inter-Process Communication (IPC) using Pipes, Shared Memory, and Message Queues Design and implement efficient and reliable inter-process communication (IPC) mechanisms using pipes, shared memory, and message queues in C to facilitate data exchange and synchronization between multiple processes within a single system. Specific Requirements: Pipe: Create and manage unidirectional and bidirectional pipes for simple data transfer between related processes. Shared Memory: Allocate and manage shared memory segments for efficient data sharing between multiple processes. Message Queues: Create and utilize message queues for asynchronous communication and data exchange with message prioritization. Synchronization: Implement appropriate synchronization mechanisms (e.g., semaphores, mutexes) to coordinate access to shared resources and prevent race conditions. Error Handling: Incorporate robust error handling to manage potential IPC failures and resource leaks.**

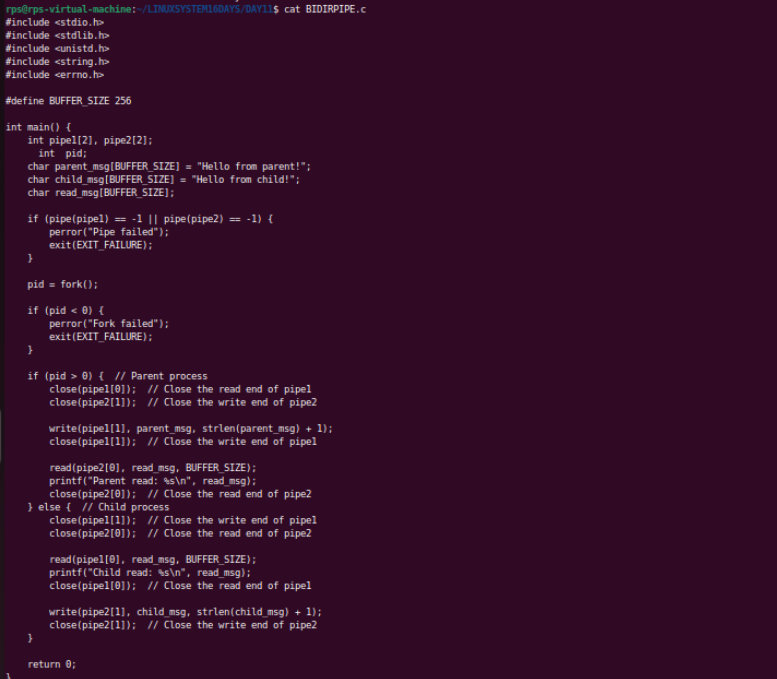
**Unidirectional Pipe**

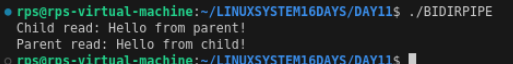
****

**Output**

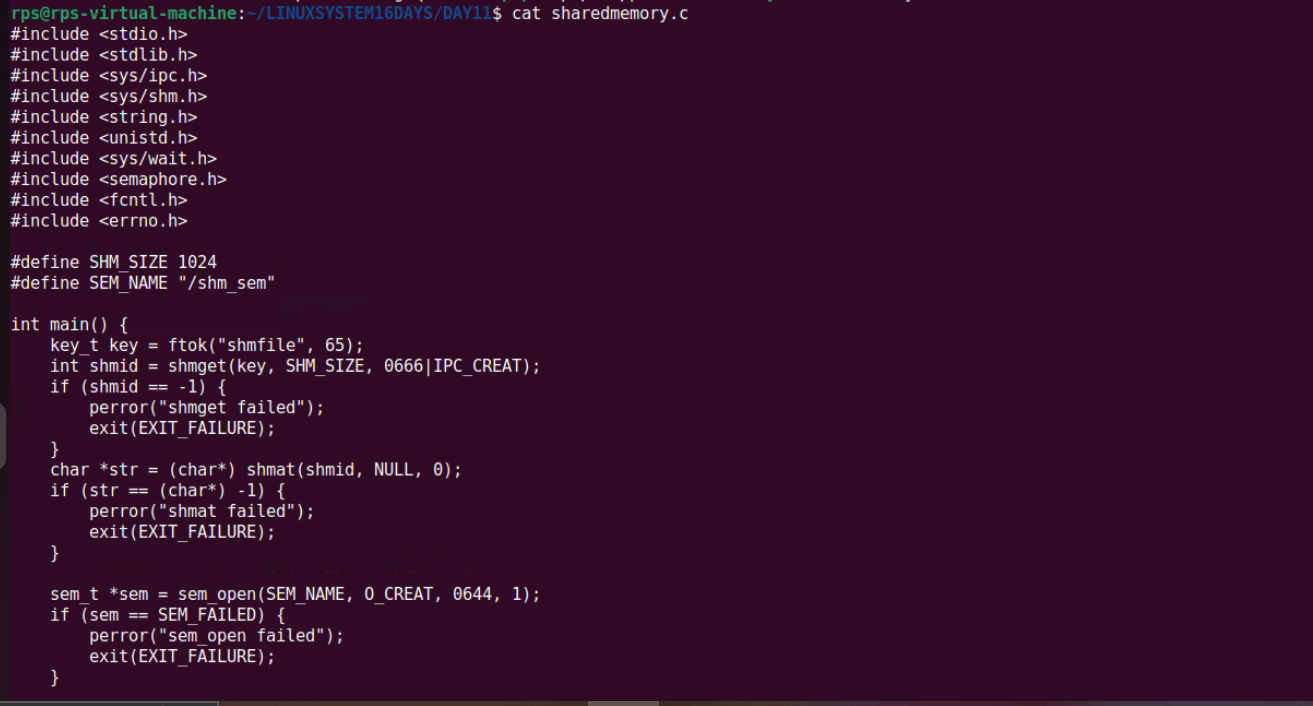
****

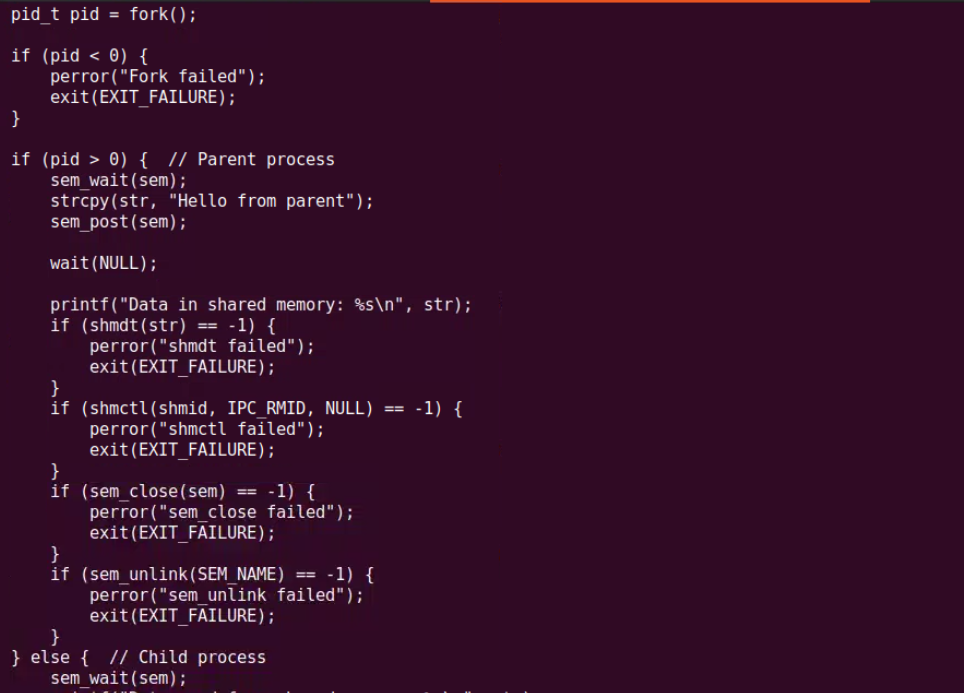
**Bidirectional Pipe**

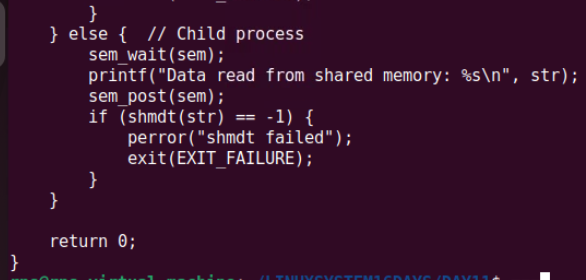
****

****

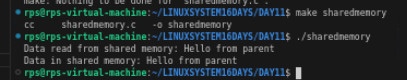
**SHARED MEMORY**

****

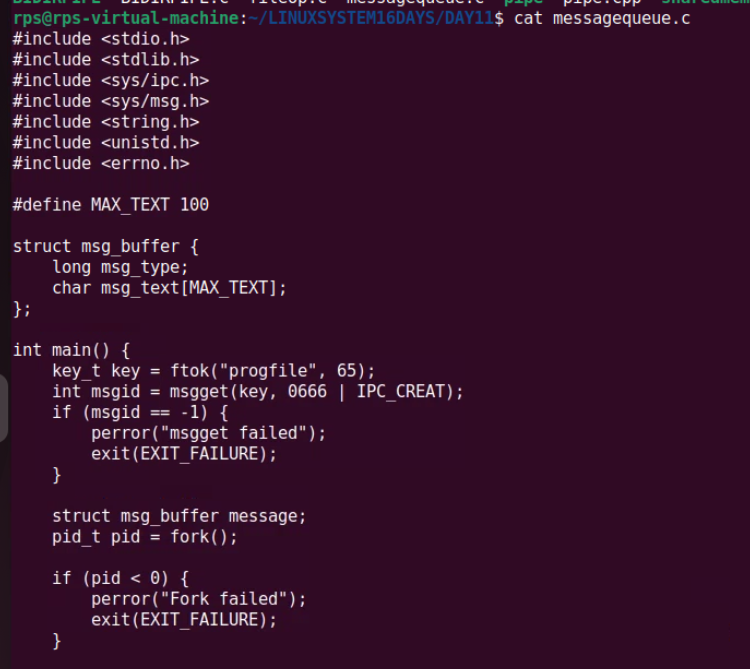
****

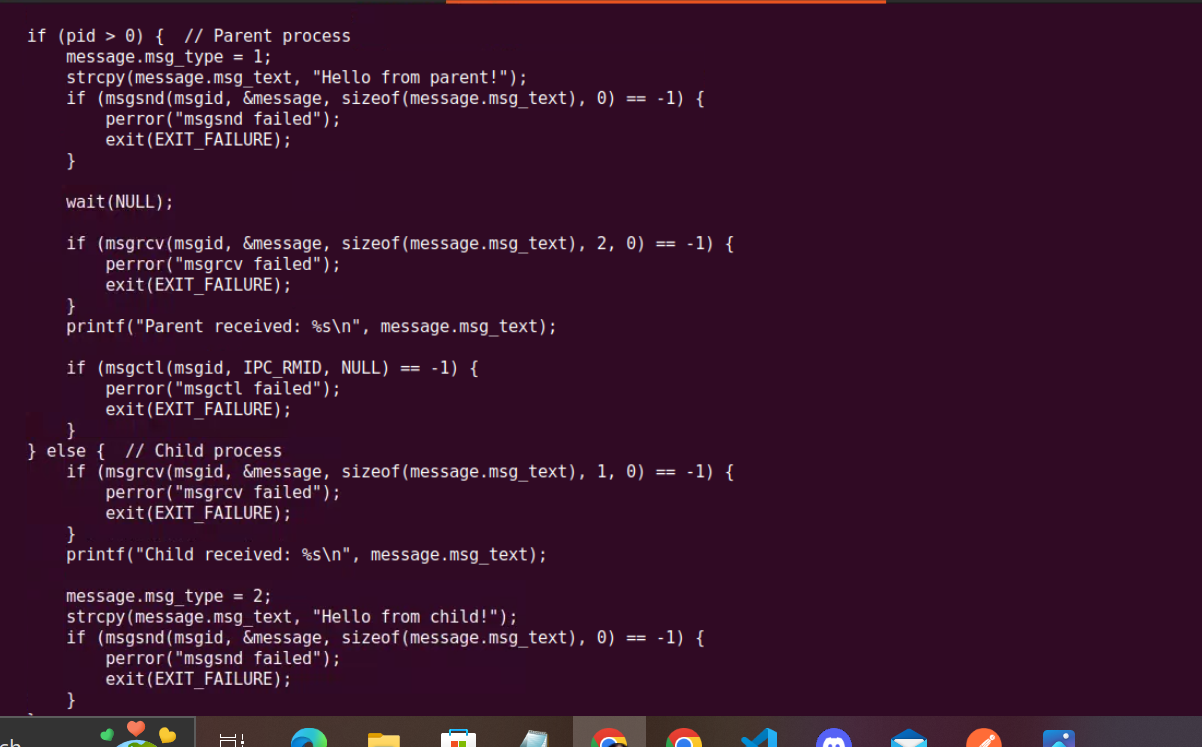
****

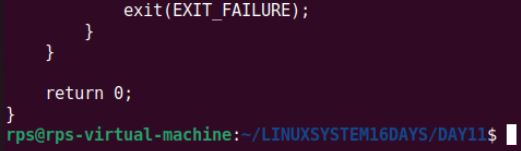
**OUTPUT**

****

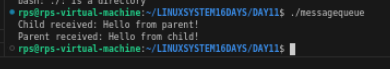
**MESSAGEQUEUE**

****

****

****

**OUTPUT**

****

**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.**

**Coding Questions in C++**

**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?**

**Sockets for Network Communication:**

**Implement a simple echo server in C++ that listens on a specific port, accepts client connections, and echoes back any messages received from clients.**

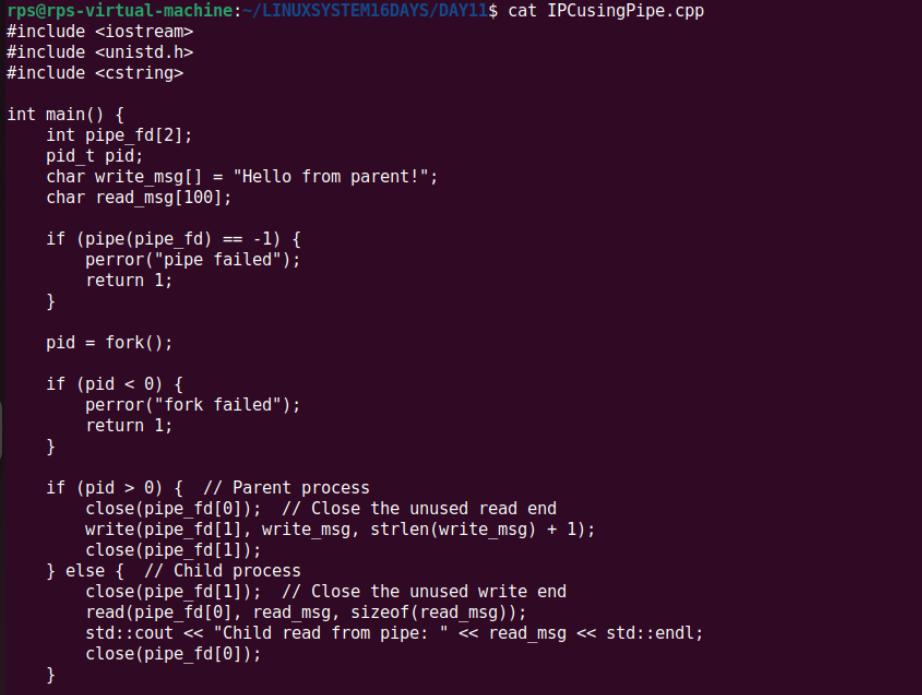
**Write a client program that connects to the echo server, sends a message, and prints the echoed response.**

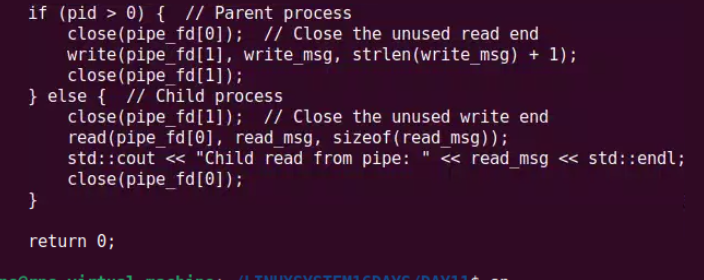
**Inter-Process Communication (IPC):**

**Write a C++ program that creates a parent process and a child process. Use a pipe for IPC to send a message from the parent to the child, and have the child process print the message.**

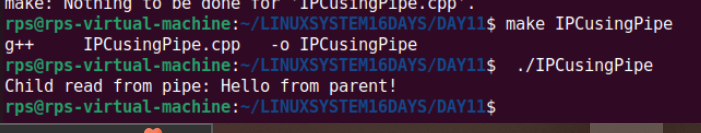
**How would you modify the program to use a message queue instead of a pipe for communication between the parent and child processes?**

**IPC INTER PROCESS COMMUNICATION**

****

****

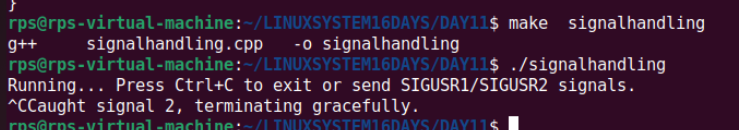
**output**

****

**SIGNAL HANDLING**

****

**OUTPUT**

****

**C++ ASSIGNMENT**

**Objective Questions (15)**

**What is the correct syntax for declaring a constant integer in C++?**

**a) const int x;**

**b) int const x;**

**c) const int x = 10;**

**d) Both a and c**

**What is the output of the following code?**

**c>**

**#include <iostream>**

**int main() {**

**int x = 10;**

**std::cout << (x > 5 ? x : 5);**

**return 0;**

**}**

**a) 5**

**b) 10**

**c) 15**

**d) 0**

**b>**

**Which of the following access specifiers allows access only within the class itself and its derived classes?**

**a) public**

**b) private**

**c) protected**

**d) friend**

**c>**

**What will be the output of the following code?**

**#include <iostream>**

**class Base {**

**public:**

**virtual void show() { std::cout << "Base"; }**

**};**

**class Derived : public Base {**

**public:**

**void show() override { std::cout << "Derived"; }**

**};**

**int main() {**

**Base\* b = new Derived();**

**b->show();**

**delete b;**

**return 0;**

**}**

**a) Base**

**b) Derived**

**c) BaseDerived**

**d) Compilation error**

**b>**

**What is the output of the following code?**

**#include <iostream>**

**int main() {**

**int arr[3] = {1, 2, 3};**

**std::cout << arr[2];**

**return 0;**

**}**

**a) 1**

**b) 2**

**c) 3**

**d) Compilation error**

**c>**

**Which keyword is used to prevent a class from being instantiated?**

**a) final**

**b) abstract**

**c) static**

**d) virtual**

**b>**

**What will be the output of the following code?**

**#include <iostream>**

**int main() {**

**int x = 5;**

**int& y = x;**

**y = 10;**

**std::cout << x;**

**return 0;**

**}**

**a) 5**

**b) 10**

**c) 15**

**d) Compilation error**

**b>**

**Which of the following is the correct way to declare a pure virtual function in C++?**

**a) virtual void func() = 0;**

**b) virtual void func();**

**c) void func() = 0;**

**d) pure virtual void func();**

**What is the output of the following code?**

**#include <iostream>**

**int main() {**

**const char\* str = "Hello";**

**str[0] = 'J';**

**std::cout << str;**

**return 0;**

**}**

**a) Hello**

**b) Jello**

**c) Compilation error**

**d) Runtime error**

**a>**

**Which operator is used to access the members of a class through a pointer?**

**a) ->**

**b) .**

**c) \***

**d) ::**

**What is the result of the following code?**

**a>**

**#include <iostream>**

**int main() {**

**int a = 5;**

**int b = 10;**

**int c = a + b;**

**std::cout << c;**

**return 0;**

**}**

**a) 5**

**b) 10**

**c) 15**

**d) Compilation error**

**c>**

**Which of the following is used to define a constant value in C++?**

**a) const**

**b) static**

**c) final**

**d) volatile**

**a>**

**What will be the output of the following code?**

**const**

**#include <iostream>**

**int main() {**

**int x = 10;**

**std::cout << ++x \* 2;**

**return 0;**

**}**

**a) 20**

**b) 22**

**c) 30**

**d) 40**

**b>**

**Which type of inheritance is also known as multiple inheritance in C++?**

**a) Single Inheritance**

**b) Multi-level Inheritance**

**c) Hierarchical Inheritance**

**d) Diamond Inheritance**

d>

**What is the output of the following code?**

**#include <iostream>**

**void func(int a, int b = 2) {**

**std::cout << a + b;**

**}**

**int main() {**

**func(3);**

**return 0;**

**}**

**a) 5**

**b) 3**

**c) 7**

**d) Compilation error**

**A>**