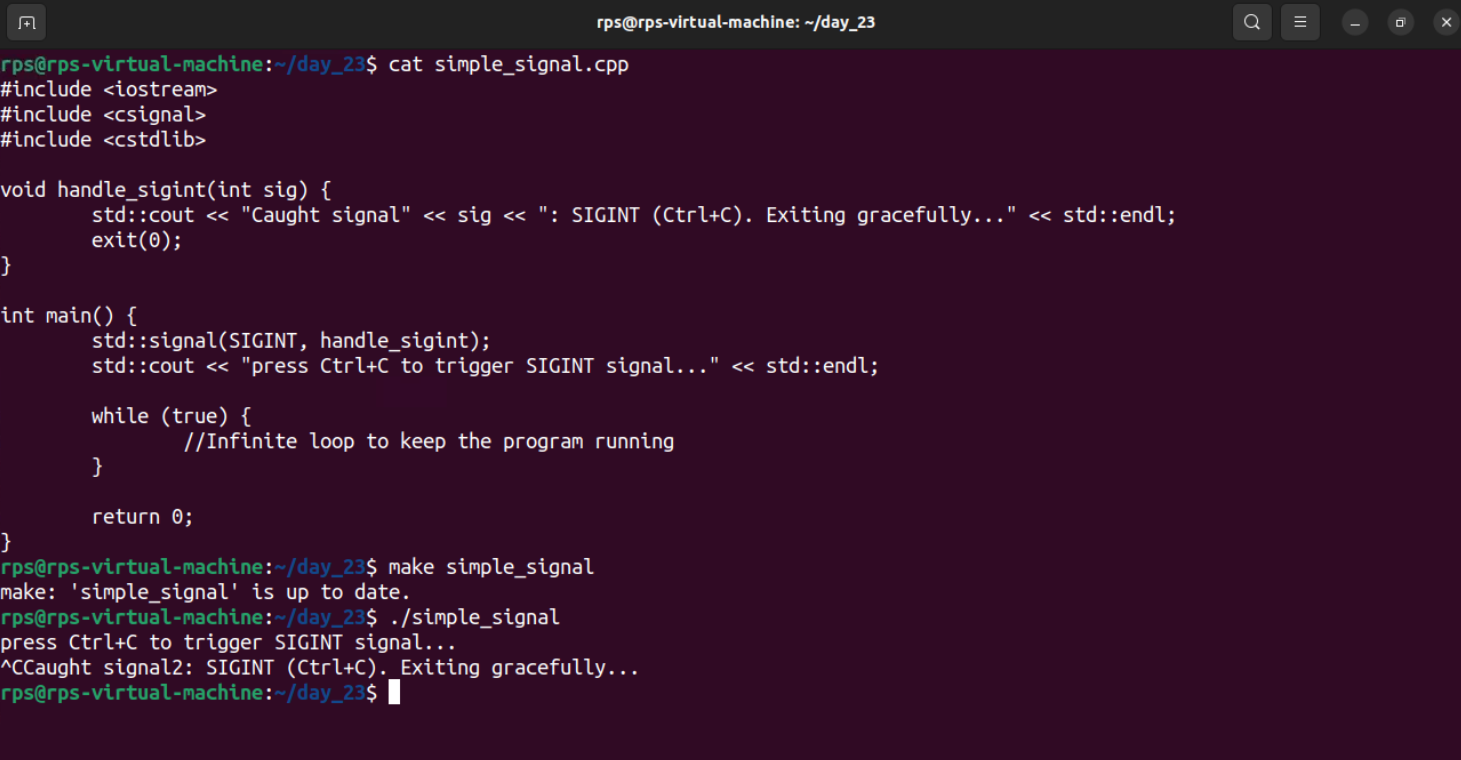
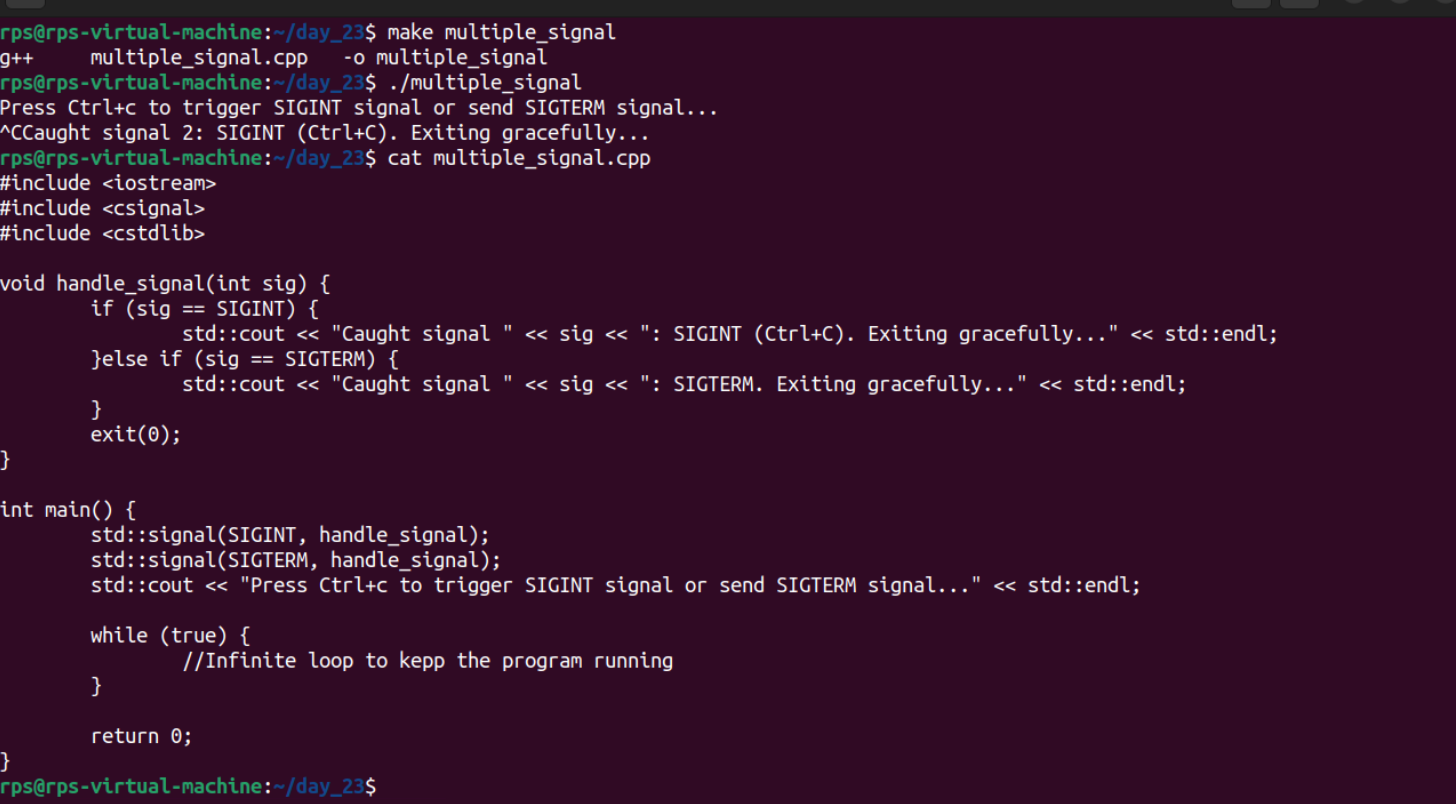
Basic Signal Handling

Simple Signal Handler: Write a C++ program that handles the SIGINT signal (Ctrl+C) gracefully by printing a custom message before exiting.

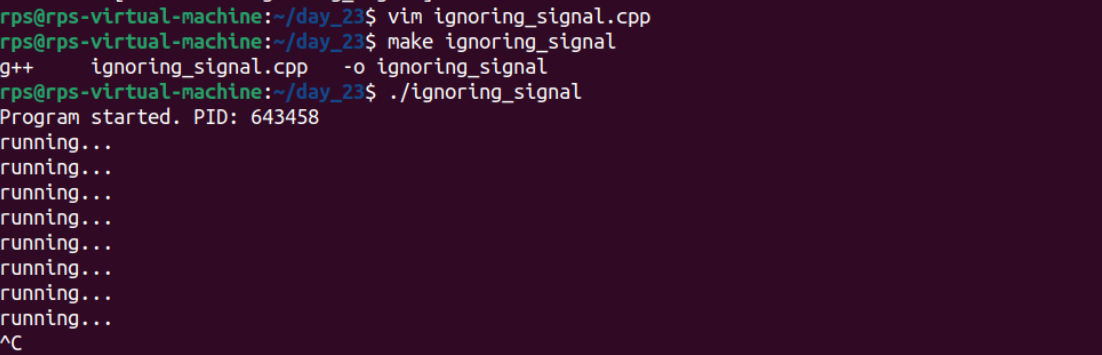


Multiple Signal Handling: Create a program that handles both SIGINT and SIGTERM signals, printing a different message for each.



Ignoring Signals: Develop a program that ignores the SIGTERM signal and continues execution even after it's sent.



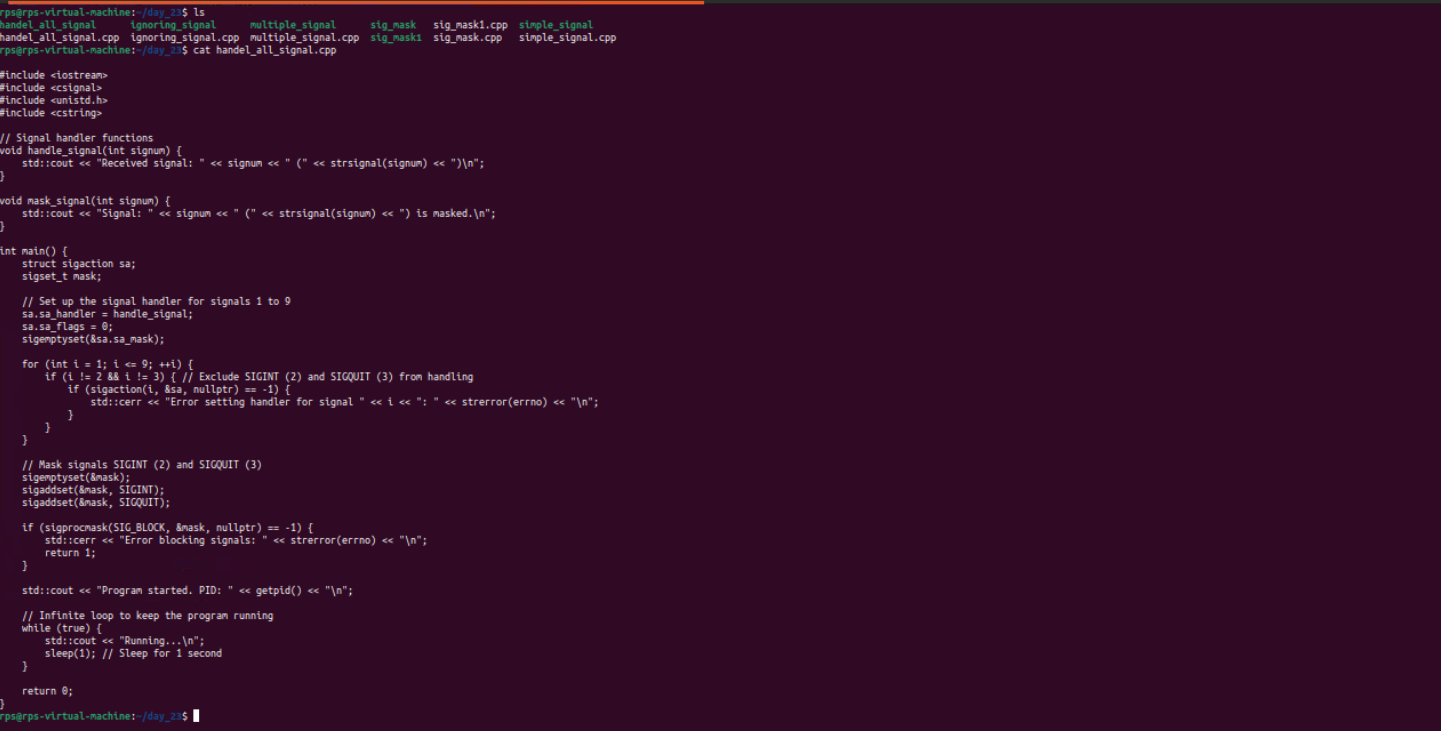


Sig\_mask

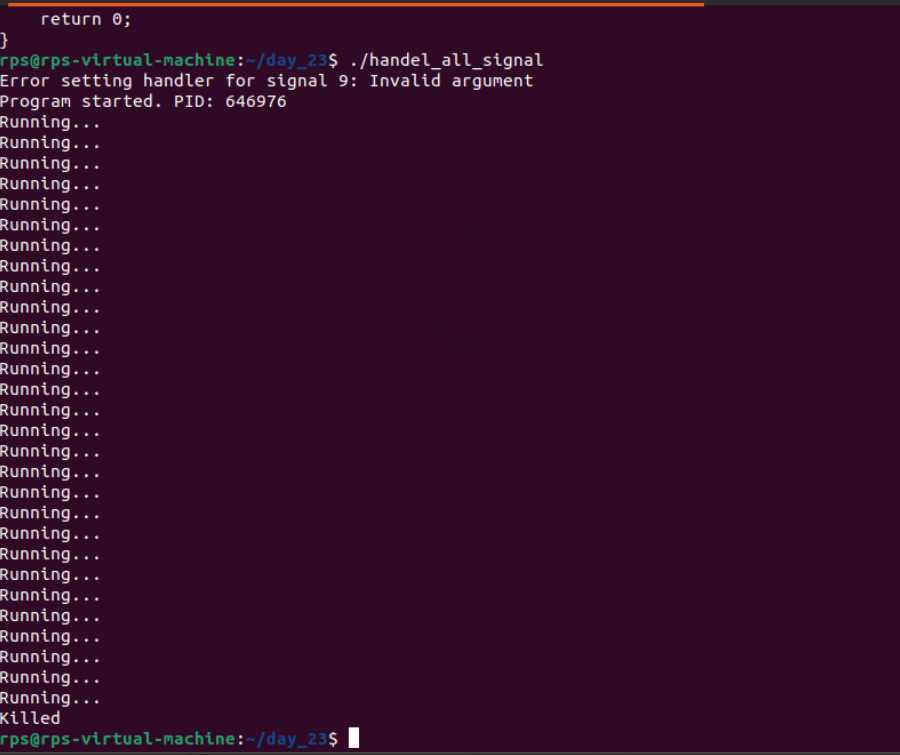
Sigterm



Code to handle the 1- 9 signal

write a code where you handle signals from 1-9 as shown and try to mask few and handle others and test 

output



Problem Statement 2: Signal Masking and Unmasking for Graceful Shutdown

Problem: Develop a C++ application that gracefully handles termination signals (e.g., SIGTERM, SIGINT) by masking specific signals during critical operations and unmasking them afterwards. Implement a clean shutdown procedure that ensures all resources are released before the process exits.

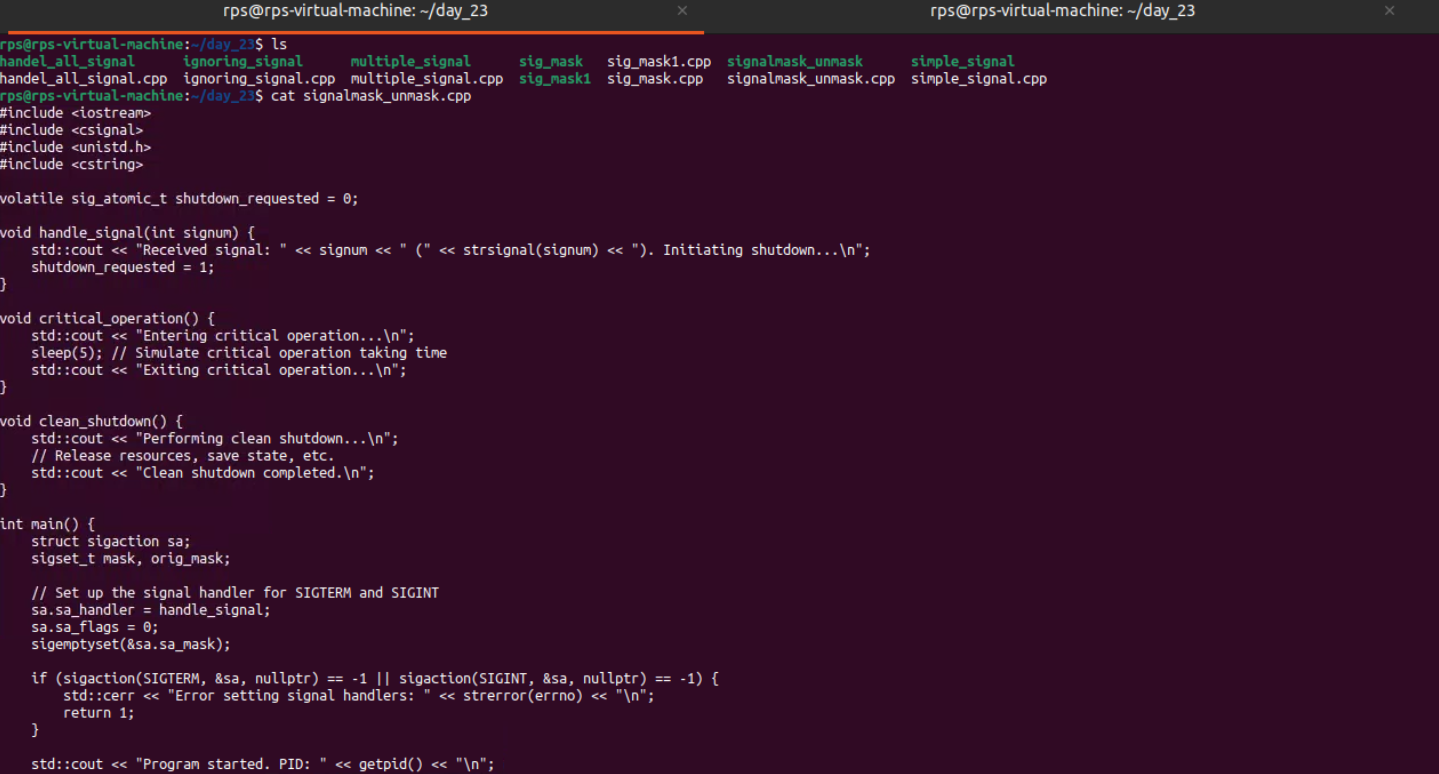
Key Challenges:

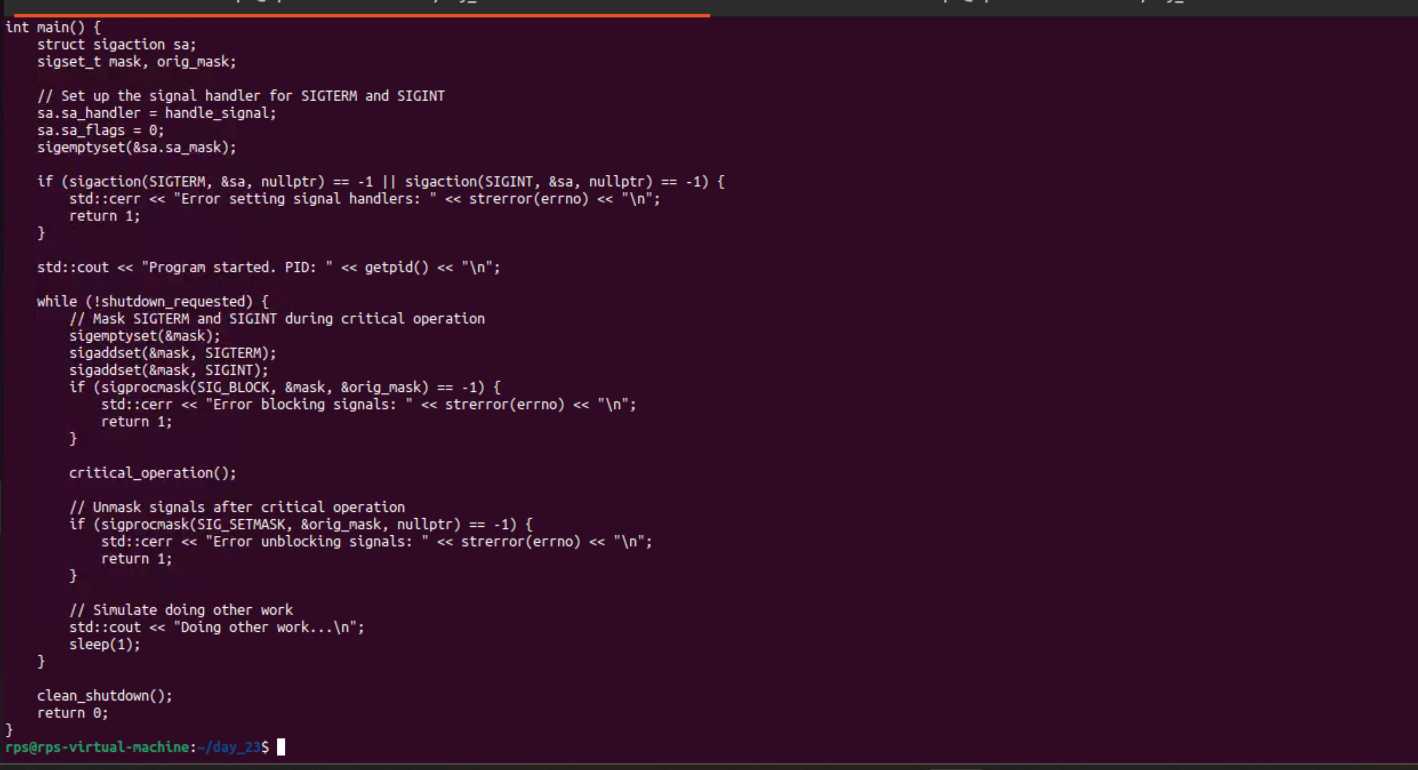
Determining the appropriate signals to mask during critical operations.

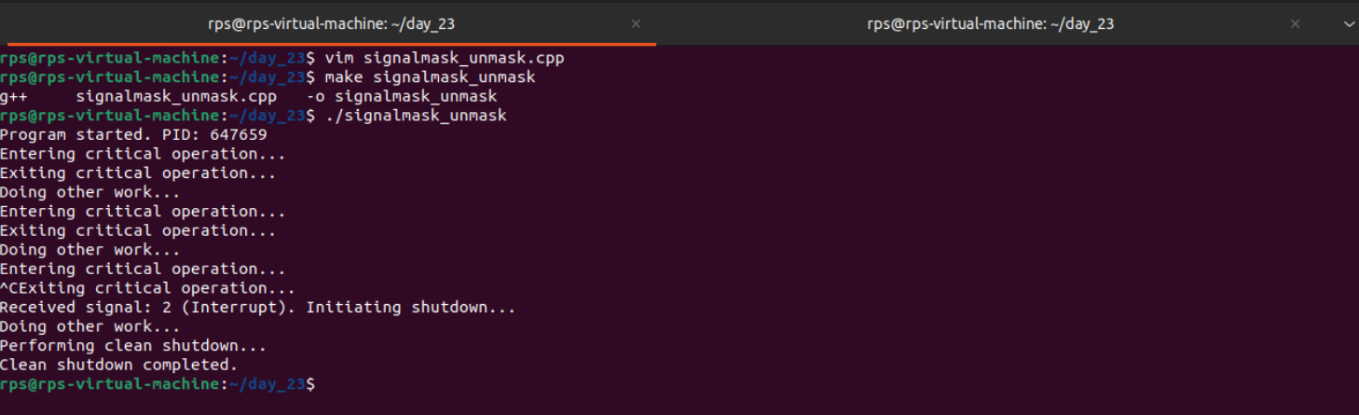
Ensuring timely unmasking of signals to avoid process hangs.

Implementing a robust shutdown mechanism that handles unexpected interruptions

Code:





Output

Problem Statement 3: Signal Masking and Unmasking for Error Handling

Problem: Create a C++ application that uses signal masking and unmasking to handle errors gracefully. Mask specific signals during error handling routines to prevent recursive signal delivery. Implement a mechanism to log error details and perform necessary cleanup actions before re-enabling the masked signals.

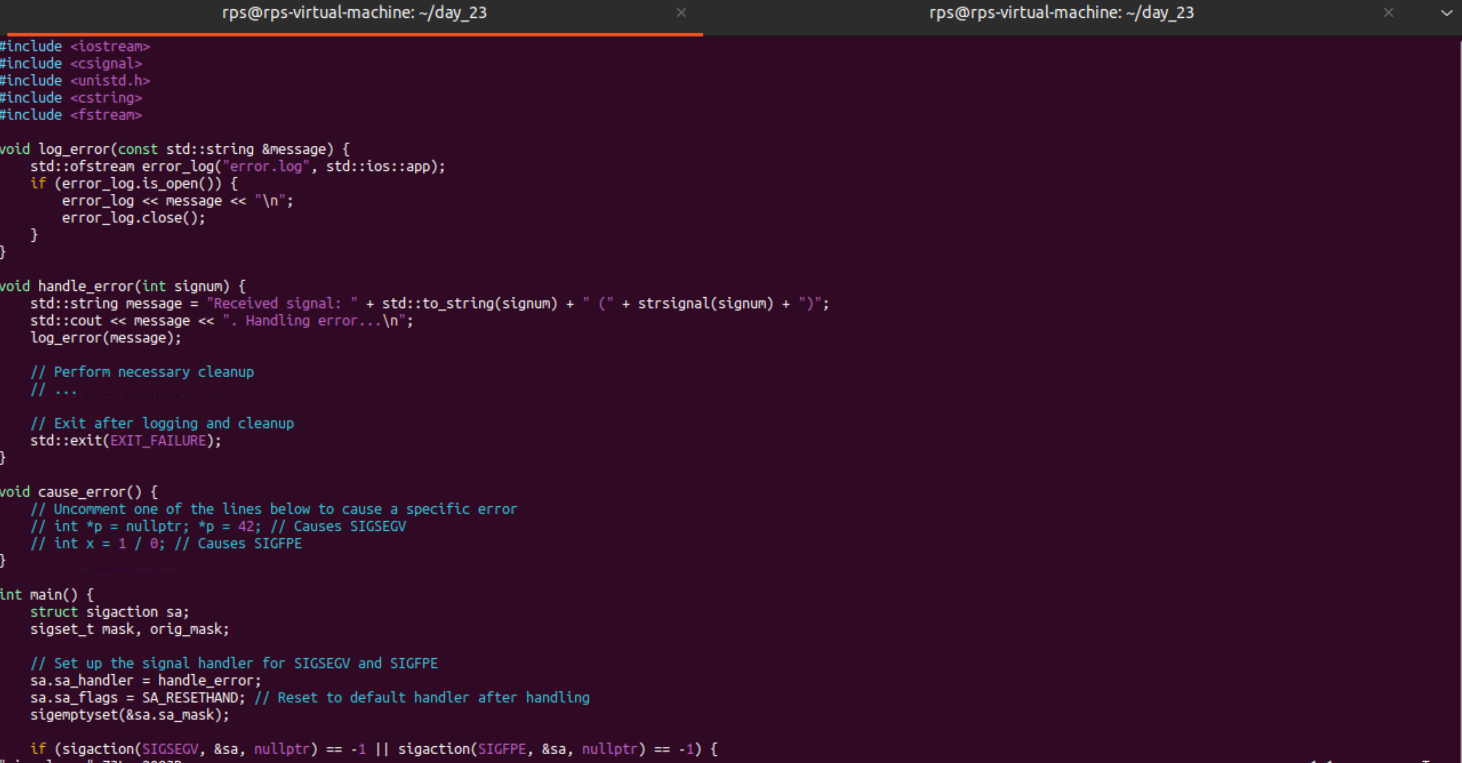
Key Challenges:

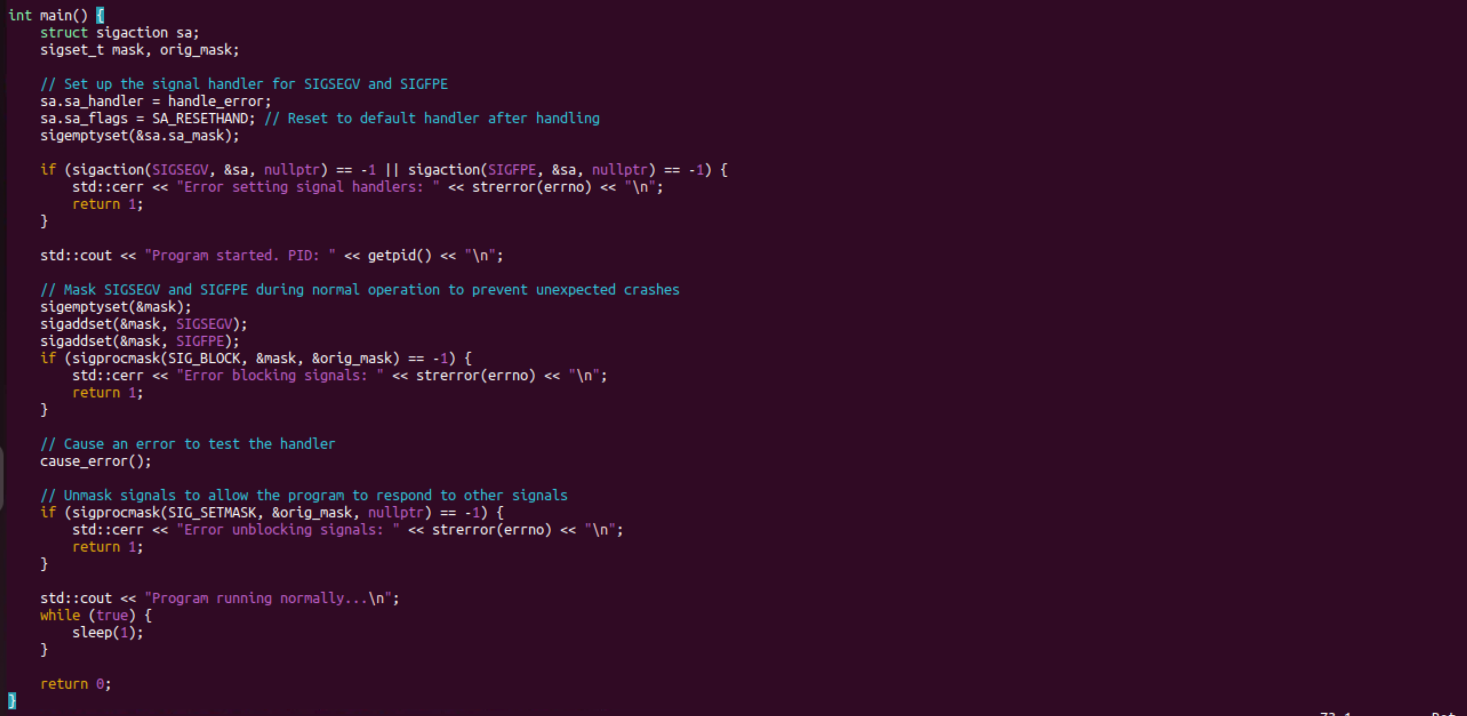
Identifying the appropriate signals to mask during error handling.

Preventing infinite recursion of signal handlers.

Ensuring proper error logging and resource cleanup.

Code:





Output

