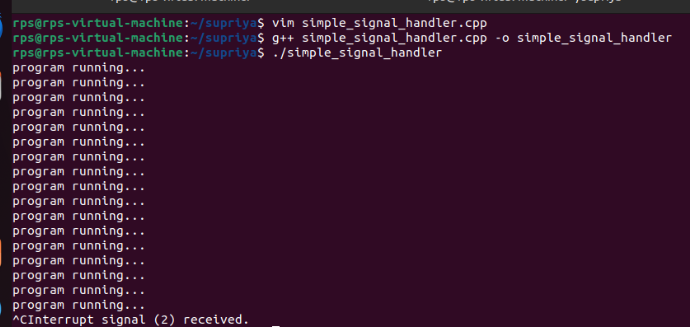
LINUX-23-07-24

TASK1: Simple Signal Handler:

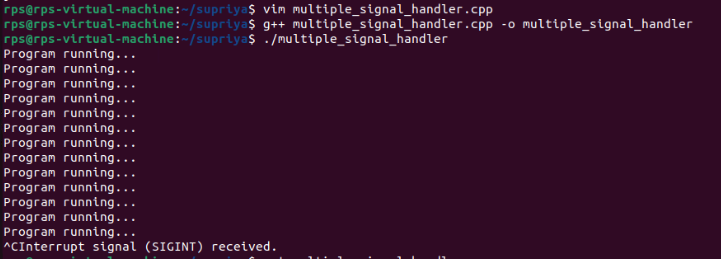
Write a C++ program that handles the SIGINT signal (Ctrl+C) gracefully by printing a custom message before exiting.

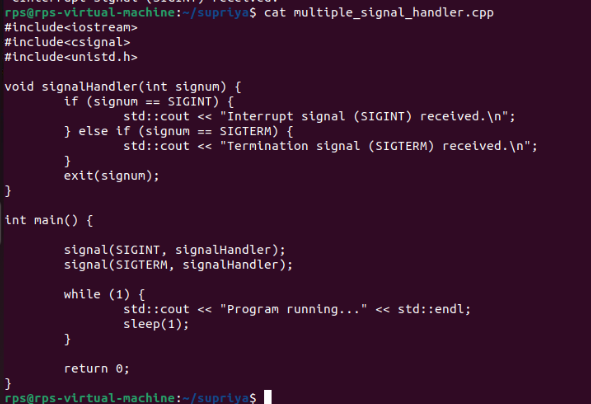


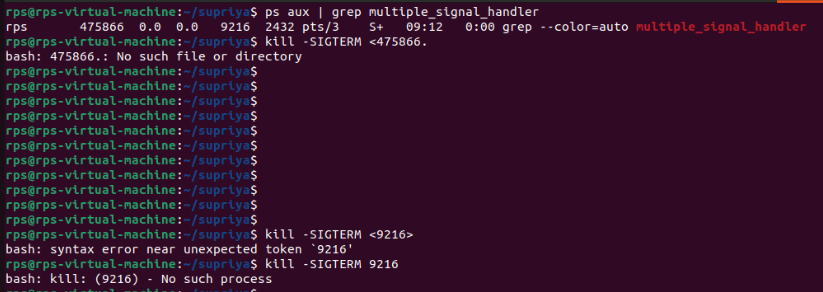


TASK2: Multiple Signal Handling:

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

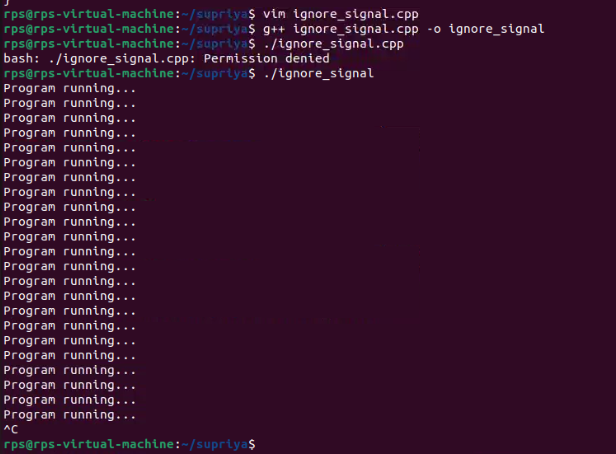




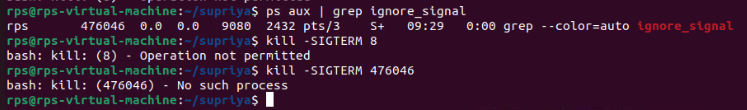


TASK3: Ignoring Signals:

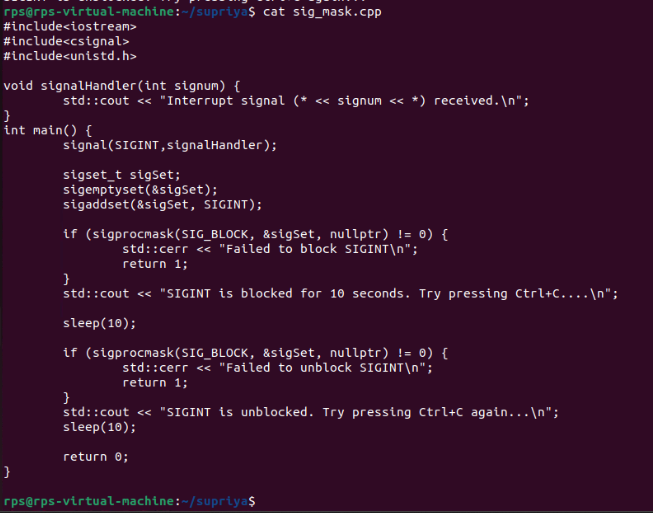
Develop a program that ignores the SIGTERM signal and continues execution even after it's sent.







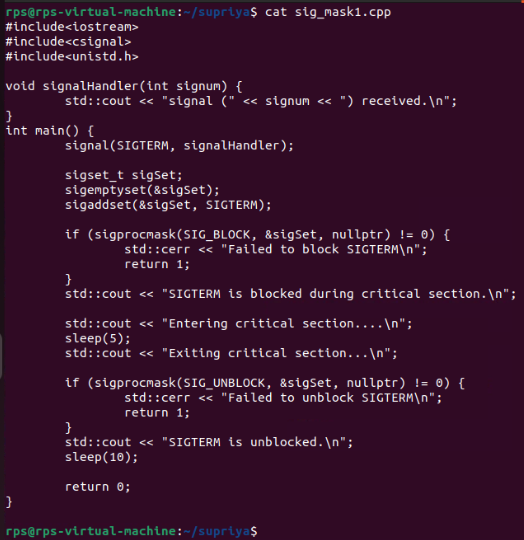
SIG\_mask code :



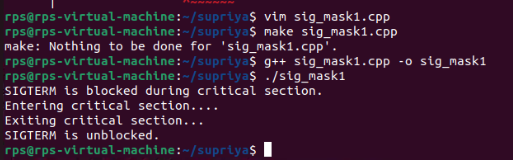
Output:



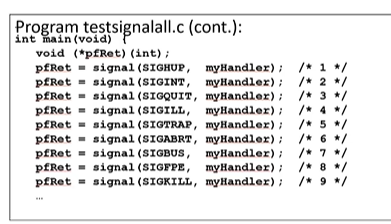
SIG\_mask1 code:

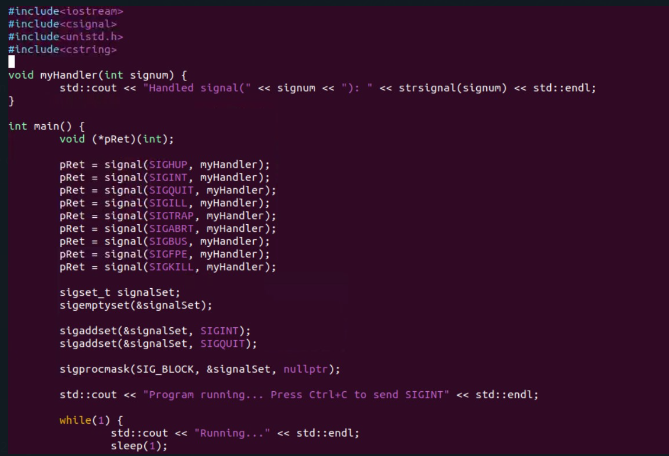


OUTPUT:



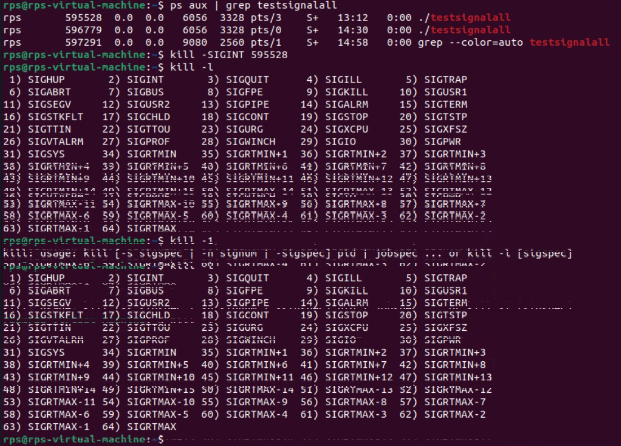
write a code where you handle signals from 1-9 as: and try to mask few and handle others and test







Output:



AFTERNOON TASKS1:

Problem Statement 2: Signal Masking and Unmasking for Graceful Shutdown

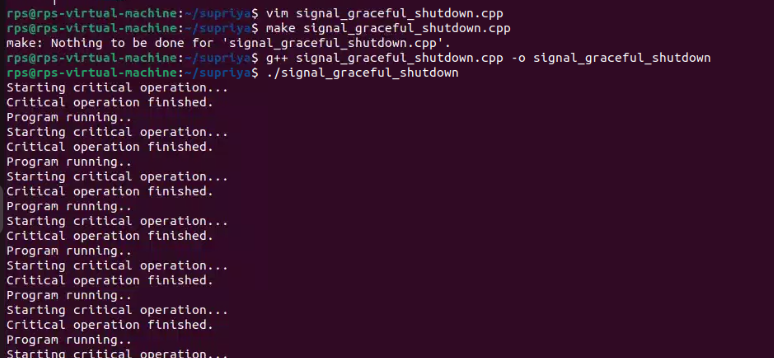
Problem: Develop a C++ application that gracefully handles termination signals (e.g., SIGTERM, SIGINT) by masking specific signals during critical operations and unmasking them afterwards. Implement a clean shutdown procedure that ensures all resources are released before the process exits.

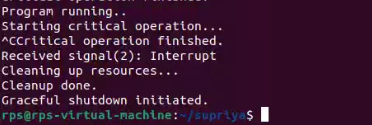
Key Challenges:

Determining the appropriate signals to mask during critical operations.

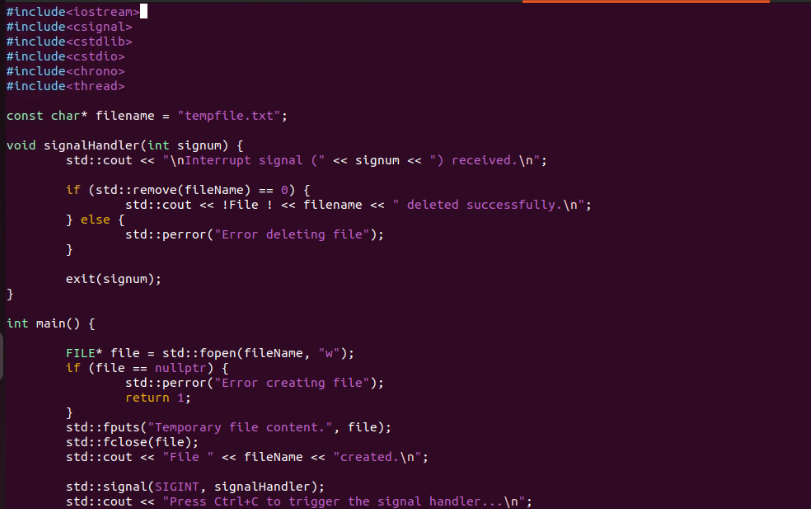
Ensuring timely unmasking of signals to avoid process hangs.

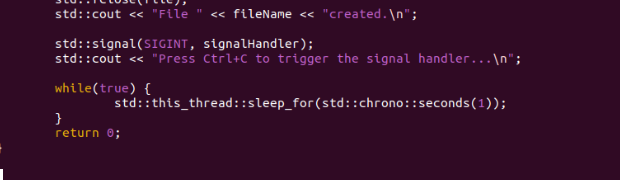
Implementing a robust shutdown mechanism that handles unexpected interruptions.





Task2:





Second code :

