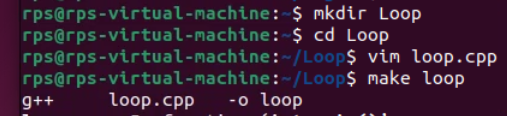
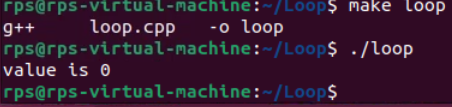
SIGNALS

TASK 1:

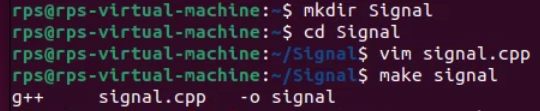
LOOP

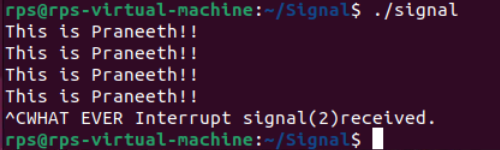




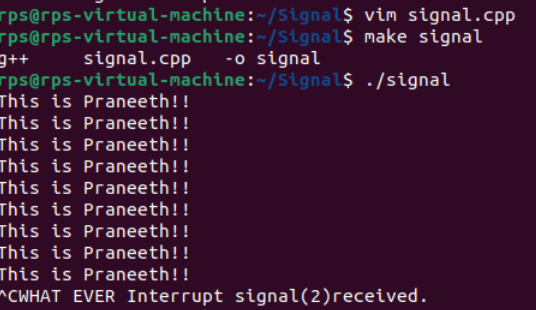
TASK 2:

SIGNAL

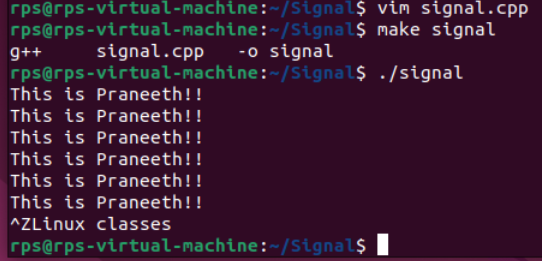




TASK 3:



TASK 4:



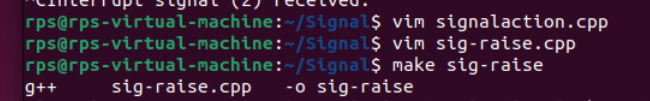
TASK 5:

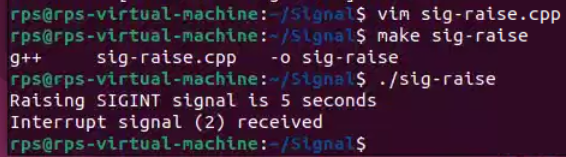
SIGNAL ACTION



TASK 6:

SIGNAL RAISE

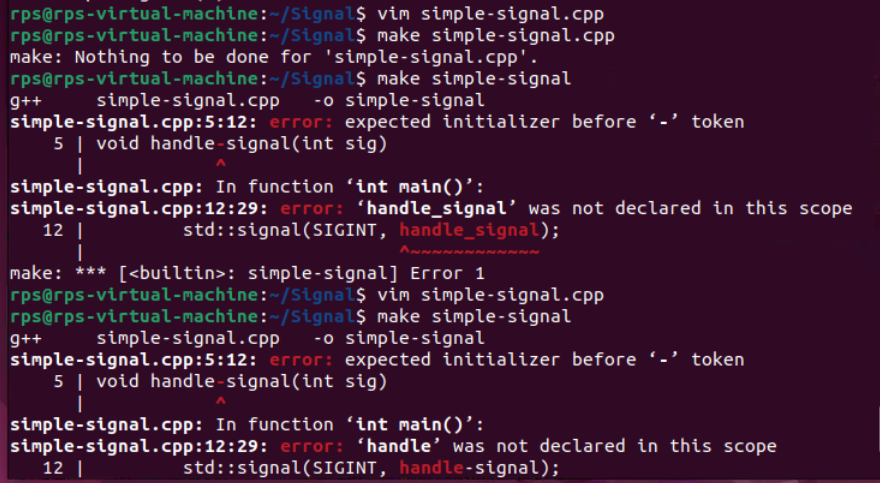


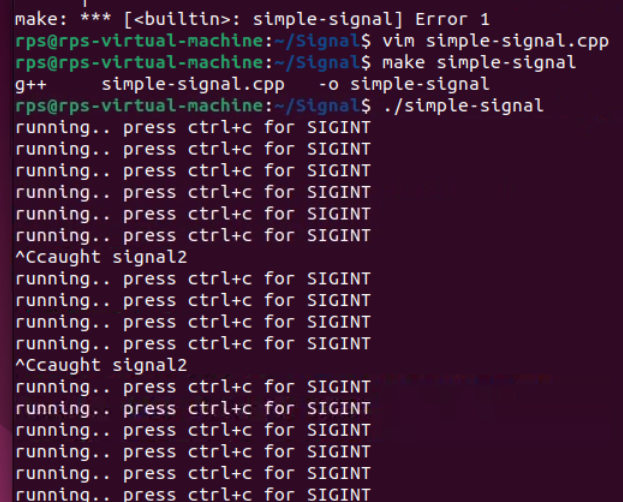


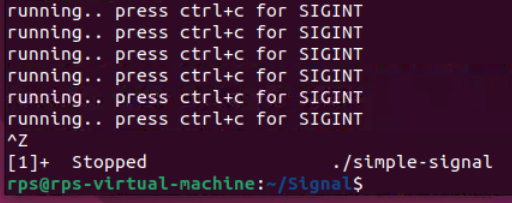
**Basic Handling vs. Advanced Control:**

**Implement signal handling using both signal and sigaction (in separate program runs). Observe the behavior. Which API allows for more control over the signal handler? Explain the key difference in a comment within your code.**

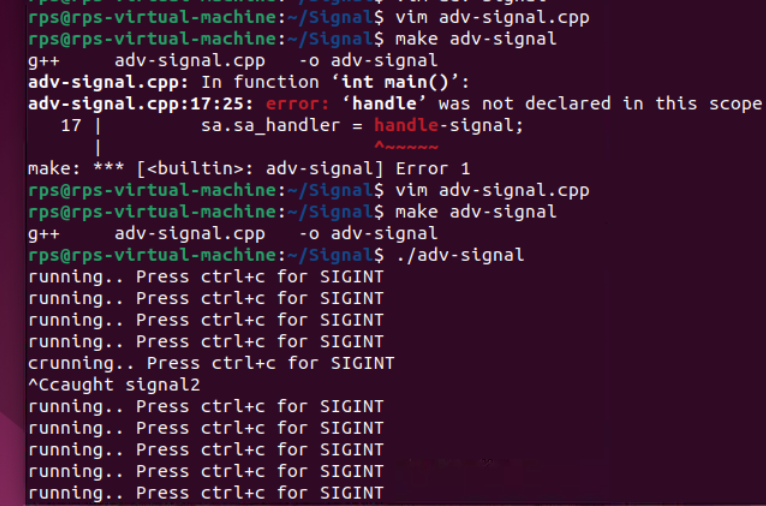
**SIMPLE Signal**

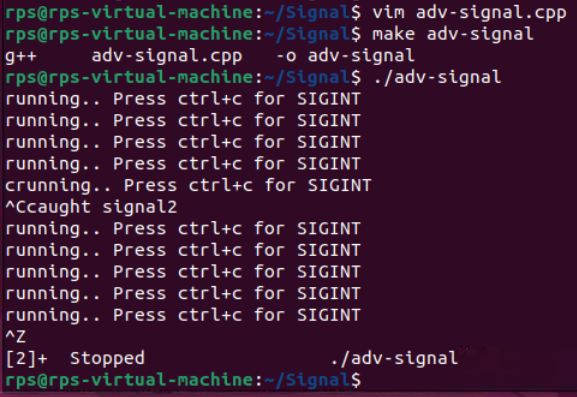
****

****

****

**ADVANCED SIGNAL**

****

****

In the **sigaction**, we use **sigfillset(&sa.sa\_mask)** to block all signals while the signal handler is running, providing more control over signal handling compared to the simpler **signal** function.

**Graceful Termination with Signal Handling**

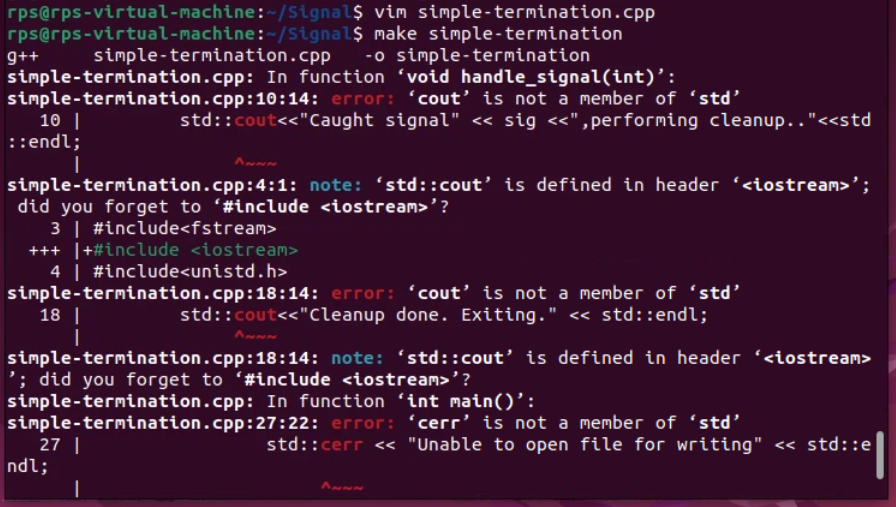
**Objective: Modify your program to demonstrate graceful termination upon receiving a specific signal (e.g., SIGINT). Within the signal handler, perform any necessary cleanup tasks (e.g., closing files, releasing resources) before exiting the program gracefully.**

**Implementation:**

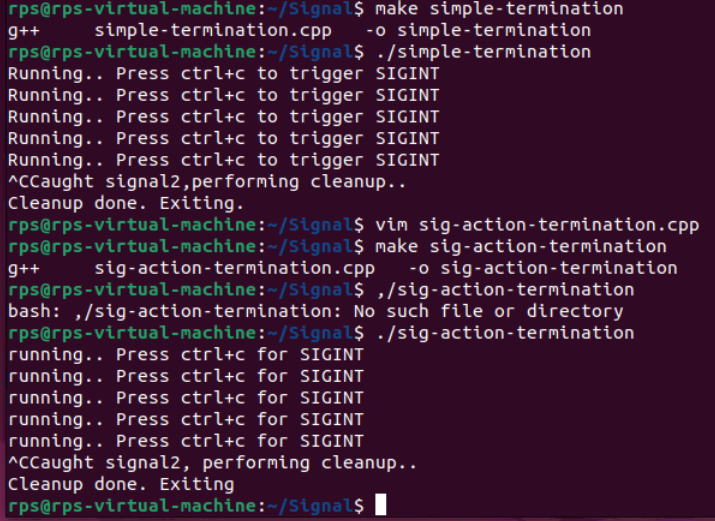
**In your signal handler function, include code to perform cleanup actions. This might involve closing open files, releasing memory, or writing data to disk.**

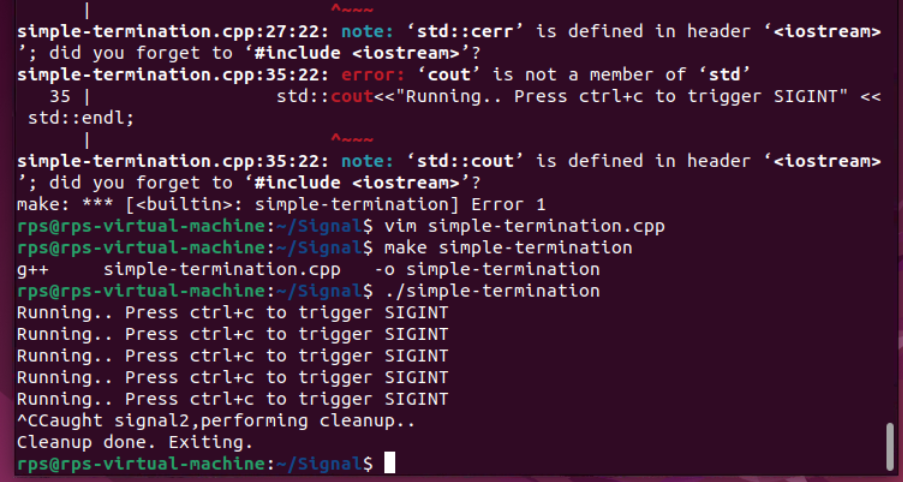
**Use exit(0) or similar methods to terminate the program after cleanup is complete.**

**SIMPLE TERMINATION**

****

**WITH SIG-ACTION-TERMINATION**

****

****