Develop a C++ application that demonstrates effective signal handling using SIGALRM, SIGDEFAULT, and SIG\_IGN. The program should:

Set up a timer using alarm() to generate a SIGALRM signal after a specified interval.

Define a signal handler function to process the SIGALRM signal and perform specific actions, such as printing a message, updating a counter, or triggering an event.

Implement logic to handle other signals (e.g., SIGINT, SIGTERM) using SIGDEFAULT or SIG\_IGN as appropriate.

Explore the behavior of the application under different signal combinations and handling strategies.

Additional Considerations:

Consider the impact of signal handling on program execution and potential race conditions.

Investigate the use of sigaction for more advanced signal handling capabilities.

Explore the application of signal handling in real-world scenarios, such as timeouts, asynchronous events, and error handling.

#include <iostream>

#include <csignal>

#include <unistd.h>

// Global counter for SIGALRM occurrences

volatile sig\_atomic\_t alarm\_counter = 0;

// Signal handler for SIGALRM

void handle\_sigalrm(int sig) {

if (sig == SIGALRM) {

alarm\_counter++;

std::cout << "SIGALRM received! Counter: " << alarm\_counter << std::endl;

// Re-arm the alarm for the next interval

alarm(2);

}

}

// Signal handler for SIGINT

void handle\_sigint(int sig) {

if (sig == SIGINT) {

std::cout << "SIGINT received! Exiting the program." << std::endl;

exit(0);

}

}

// Function to set up signal handlers using sigaction

void setup\_signal\_handlers() {

struct sigaction sa;

// Set up handler for SIGALRM

sa.sa\_handler = handle\_sigalrm;

sa.sa\_flags = 0;

sigemptyset(&sa.sa\_mask);

sigaction(SIGALRM, &sa, nullptr);

// Set up handler for SIGINT

sa.sa\_handler = handle\_sigint;

sigaction(SIGINT, &sa, nullptr);

// Ignore SIGTERM

sa.sa\_handler = SIG\_IGN;

sigaction(SIGTERM, &sa, nullptr);

}

int main() {

// Set up signal handlers

setup\_signal\_handlers();

// Set an alarm for 2 seconds

alarm(2);

// Main loop

while (true) {

std::cout << "Running... Press Ctrl+C to exit." << std::endl;

sleep(1);

}

return 0;

}