**Signal masking and unmasking**

#include <iostream>

#include <csignal>

#include <unistd.h>

#include <cstring>

volatile sig\_atomic\_t shutdown\_requested = 0;

void handle\_signal(int signum) {

std::cout << "Received signal: " << signum << " (" << strsignal(signum) << "). Initiating shutdown...\n";

shutdown\_requested = 1;

}

void critical\_operation() {

std::cout << "Entering critical operation...\n";

sleep(5); // Simulate critical operation taking time

std::cout << "Exiting critical operation...\n";

}

void clean\_shutdown() {

std::cout << "Performing clean shutdown...\n";

// Release resources, save state, etc.

std::cout << "Clean shutdown completed.\n";

}

int main() {

struct sigaction sa;

sigset\_t mask, orig\_mask;

// Set up the signal handler for SIGTERM and SIGINT

sa.sa\_handler = handle\_signal;

sa.sa\_flags = 0;

sigemptyset(&sa.sa\_mask);

if (sigaction(SIGTERM, &sa, nullptr) == -1 || sigaction(SIGINT, &sa, nullptr) == -1) {

std::cerr << "Error setting signal handlers: " << strerror(errno) << "\n";

return 1;

}

std::cout << "Program started. PID: " << getpid() << "\n";

while (!shutdown\_requested) {

// Mask SIGTERM and SIGINT during critical operation

sigemptyset(&mask);

sigaddset(&mask, SIGTERM);

sigaddset(&mask, SIGINT);

if (sigprocmask(SIG\_BLOCK, &mask, &orig\_mask) == -1) {

std::cerr << "Error blocking signals: " << strerror(errno) << "\n";

return 1;

}

critical\_operation();

// Unmask signals after critical operation

if (sigprocmask(SIG\_SETMASK, &orig\_mask, nullptr) == -1) {

std::cerr << "Error unblocking signals: " << strerror(errno) << "\n";

return 1;

}

// Simulate doing other work

std::cout << "Doing other work...\n";

sleep(1);

}

clean\_shutdown();

return 0;

}