**Signal masking and unmasking for error handling**

#include <iostream>

#include <csignal>

#include <unistd.h>

#include <cstring>

#include <fstream>

void log\_error(const std::string &message) {

std::ofstream error\_log("error.log", std::ios::app);

if (error\_log.is\_open()) {

error\_log << message << "\n";

error\_log.close();

}

}

void handle\_error(int signum) {

std::string message = "Received signal: " + std::to\_string(signum) + " (" + strsignal(signum) + ")";

std::cout << message << ". Handling error...\n";

log\_error(message);

// Perform necessary cleanup

// ...

// Exit after logging and cleanup

std::exit(EXIT\_FAILURE);

}

void cause\_error() {

// Uncomment one of the lines below to cause a specific error

// int \*p = nullptr; \*p = 42; // Causes SIGSEGV

// int x = 1 / 0; // Causes SIGFPE

}

int main() {

struct sigaction sa;

sigset\_t mask, orig\_mask;

// Set up the signal handler for SIGSEGV and SIGFPE

sa.sa\_handler = handle\_error;

sa.sa\_flags = SA\_RESETHAND; // Reset to default handler after handling

sigemptyset(&sa.sa\_mask);

if (sigaction(SIGSEGV, &sa, nullptr) == -1 || sigaction(SIGFPE, &sa, nullptr) == -1) {

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

return 1;

}

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

// Mask SIGSEGV and SIGFPE during normal operation to prevent unexpected crashes

sigemptyset(&mask);

sigaddset(&mask, SIGSEGV);

sigaddset(&mask, SIGFPE);

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

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

return 1;

}

// Cause an error to test the handler

cause\_error();

// Unmask signals to allow the program to respond to other signals

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

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

return 1;

}

std::cout << "Program running normally...\n";

while (true) {

sleep(1);

}

return 0;

}