Створіть фреймворк, який дозволяє будь-якому додатку підключити власний логгер падіння (core-dump обробник), не впливаючи на основну логіку.

Main.c

#include <stdio.h>

#include "crash\_logger.h"

void my\_crash\_callback(int signo) {

    // Додаткові дії при падінні, наприклад, запис у файл

    FILE \*f = fopen("crash\_log.txt", "a");

    if (f) {

        fprintf(f, "Crash callback: signal %d\n", signo);

        fclose(f);

    }

}

int main() {

    crash\_logger\_init(my\_crash\_callback);

    printf("App started\n");

    // Примусове падіння, щоб перевірити логер

    int \*p = NULL;

    \*p = 42;  // викликає SIGSEGV

    return 0;

}

crash\_logger.h

#ifndef CRASH\_LOGGER\_H

#define CRASH\_LOGGER\_H

#ifdef \_\_cplusplus

extern "C" {

#endif

typedef void (\*crash\_handler\_callback\_t)(int signo);

/// Ініціалізувати логер падінь.

/// При переданні callback - він буде викликаний під час падіння.

void crash\_logger\_init(crash\_handler\_callback\_t callback);

#ifdef \_\_cplusplus

}

#endif

#endif // CRASH\_LOGGER\_H

crash\_logger.c

#define \_GNU\_SOURCE

#include <signal.h>

#include <execinfo.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include "crash\_logger.h"

#include <string.h>

static crash\_handler\_callback\_t user\_callback = NULL;

static void signal\_handler(int signo) {

    void \*buffer[30];

    int size;

    // Вивід сигналу

    fprintf(stderr, "Crash detected: signal %d (%s)\n", signo, strsignal(signo));

    // Стек викликів

    size = backtrace(buffer, 30);

    fprintf(stderr, "Stack trace:\n");

    backtrace\_symbols\_fd(buffer, size, STDERR\_FILENO);

    // Виклик користувацького callback, якщо задано

    if (user\_callback) {

        user\_callback(signo);

    }

    // Відновлення поведінки за замовчуванням і повторне відправлення сигналу,

    // щоб ядро могло створити core dump.

    signal(signo, SIG\_DFL);

    raise(signo);

}

void crash\_logger\_init(crash\_handler\_callback\_t callback) {

    user\_callback = callback;

    // Сигнали для обробки падіння

    int signals[] = {SIGSEGV, SIGABRT, SIGFPE, SIGILL, SIGBUS};

    int n = sizeof(signals) / sizeof(signals[0]);

    for (int i = 0; i < n; i++) {

        struct sigaction sa;

        sa.sa\_handler = signal\_handler;

        sigemptyset(&sa.sa\_mask);

        sa.sa\_flags = SA\_RESTART | SA\_NODEFER;

        sigaction(signals[i], &sa, NULL);

    }

}



