## Prompt

Đây là lỗi khi chạy code C: Segmentation fault (core dumped). Hãy phân tích nguyên nhân và gợi ý sửa lỗi.  
  
Code gốc:  
  
#include <stdio.h>  
  
int main() {  
 int \*p = NULL; // không cấp phát bộ nhớ, p trỏ tới NULL  
 \*p = 42; // ghi vào địa chỉ NULL -> segmentation fault  
 printf("Value: %d\n", \*p);  
 return 0;  
}

## Code gốc

#include <stdio.h>  
  
int main() {  
 int \*p = NULL; // không cấp phát bộ nhớ, p trỏ tới NULL  
 \*p = 42; // ghi vào địa chỉ NULL -> segmentation fault  
 printf("Value: %d\n", \*p);  
 return 0;  
}

## Kết quả khi biên dịch và chạy

Biên dịch (gcc):

(Không có lỗi biên dịch)

Kết quả chạy chương trình:

Return code: -11

STDOUT:

(Không có đầu ra STDOUT)

STDERR:

(Không có đầu ra STDERR)

## Phân tích lỗi và gợi ý sửa (phân tích của AI)

Phân tích nguyên nhân lỗi:  
- Nguyên nhân trực tiếp: chương trình ghi vào vị trí nhớ qua con trỏ NULL (int \*p = NULL; \*p = 42;).  
 Việc dereference (truy cập) con trỏ NULL không hợp lệ và hệ điều hành sẽ ngắt tiến trình -> "Segmentation fault".  
- Điểm cần kiểm tra chung: con trỏ chưa được cấp phát bộ nhớ hoặc chưa được gán địa chỉ hợp lệ trước khi sử dụng.  
- Các nguyên nhân khác (không có trong ví dụ này nhưng thường gặp): truy cập mảng ngoài phạm vi, dùng con trỏ đã bị free, chia cho 0 khi kết hợp với con trỏ, v.v.  
  
Gợi ý sửa lỗi:  
1. Cấp phát bộ nhớ cho con trỏ trước khi ghi:  
 - Dùng malloc để cấp phát vùng nhớ, hoặc gán con trỏ tới một biến hợp lệ.  
2. Sử dụng biến tĩnh/thông thường thay vì con trỏ nếu không cần thiết.  
3. Kiểm tra con trỏ khác NULL trước khi dereference: if (p != NULL) { \*p = ... }  
4. Với mảng: kiểm tra chỉ số trước khi truy cập để tránh out-of-bounds.  
  
Ví dụ sửa (cách 1: dùng biến bình thường):  
#include <stdio.h>  
  
int main() {  
 int value = 42;  
 int \*p = &value; // p trỏ tới biến hợp lệ  
 \*p = 42;  
 printf("Value: %d\n", \*p);  
 return 0;  
}  
  
Ví dụ sửa (cách 2: cấp phát động bằng malloc):  
#include <stdio.h>  
#include <stdlib.h>  
  
int main() {  
 int \*p = malloc(sizeof(int)); // cấp phát bộ nhớ  
 if (p == NULL) { // kiểm tra cấp phát thành công  
 fprintf(stderr, "Không thể cấp phát bộ nhớ\n");  
 return 1;  
 }  
 \*p = 42;  
 printf("Value: %d\n", \*p);  
 free(p); // giải phóng bộ nhớ  
 return 0;  
}