**Scenario: Segmentation Fault / Null Pointer Debugging**

**Summary:**  
Detected and fixed a segmentation fault caused by a null pointer in a C program using GDB.

**Problem Description**

The program crashed because it attempted to write to a null pointer. In real-world applications, this could cause a system crash or data corruption if not handled properly.

**Tools & Languages**

* **Language:** C
* **Debugging Tools:** GDB

**Initial Code**

#include <stdio.h>

#include <stdlib.h>

int main() {

int \*ptr = NULL; // pointer intentionally set to NULL

int n = 5;

for (int i = 0; i < n; i++) {

ptr[i] = i \* 3; // crash occurs here (segmentation fault)

printf("ptr[%d] = %d\n", i, ptr[i]);

}

return 0;

}

**Issue Detected:**

* Program crashed at line 8 with a **segmentation fault**
* GDB confirmed ptr = 0x0 (null pointer)

**Fixed Code**

#include <stdio.h>

#include <stdlib.h>

int main() {

int \*ptr;

int n = 5;

ptr = (int \*)malloc(n \* sizeof(int)); // allocate memory

if (!ptr) { // sanity check

printf("Memory allocation failed!\n");

return 1;

}

for (int i = 0; i < n; i++) {

ptr[i] = i \* 3;

printf("ptr[%d] = %d\n", i, ptr[i]);

}

free(ptr); // free memory

return 0;

}

**Fix Explanation:**

* Allocated memory before using the pointer
* Added a check for failed allocation
* Freed memory to prevent leaks

**Outcome**

Program output:

ptr[0] = 0

ptr[1] = 3

ptr[2] = 6

ptr[3] = 9

ptr[4] = 12

* Runs without segmentation fault
* Memory safely freed

**Lessons Learned**

* Null pointer dereferencing is a common source of segmentation faults
* Always allocate memory before dereferencing pointers
* Include sanity checks for memory allocation failures
* GDB is essential to pinpoint the exact line causing the crash