Module.1

Use Case 1: Debugging Logical Errors in Loops with Reverse Debugging

Overview

Logical errors in loops can produce incorrect results, such as exceeding bounds or miscalculating outputs. Reverse debugging in GDB allows you to step backward to trace how the loop state evolved.

**Logical Error:- logical\_error.c**

#include <stdio.h>

int main(){

    int n = 5;  //calculate factorial of 5

    int factorial = 1;

    //logical error: Incorrect initialization of the loop variable

    for(int i = 0; i <= n; i++){

        factorial \*= i;   //causes multiplication by 0 in the first iteration

    }

    printf("Factorial of %d is %d\n", n, factorial);

    return 0;

}

**Steps to debug with gdb to find logical error:-**

gcc -g -o logical\_error logical\_error.c

gdb ./logical\_error

break main

target record-full

run

next

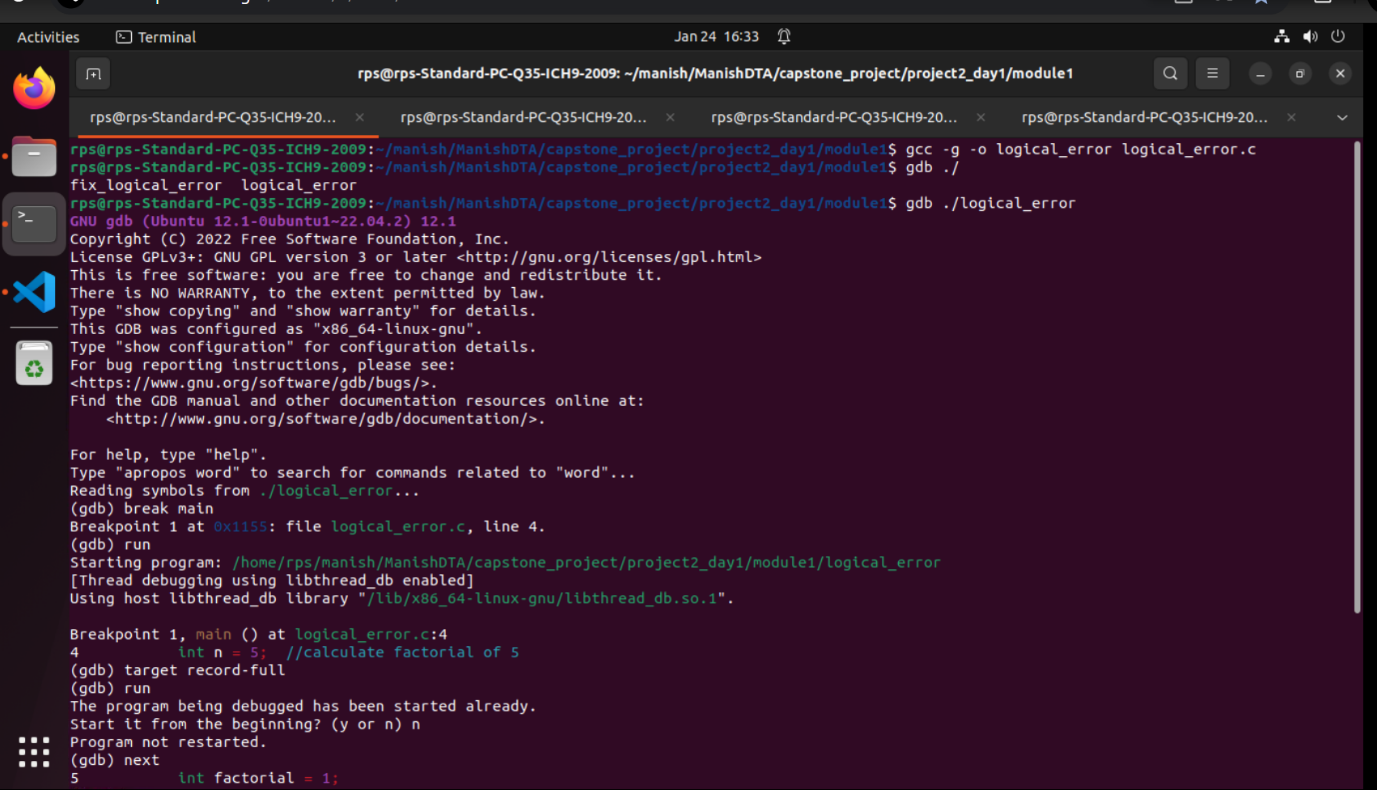
print i

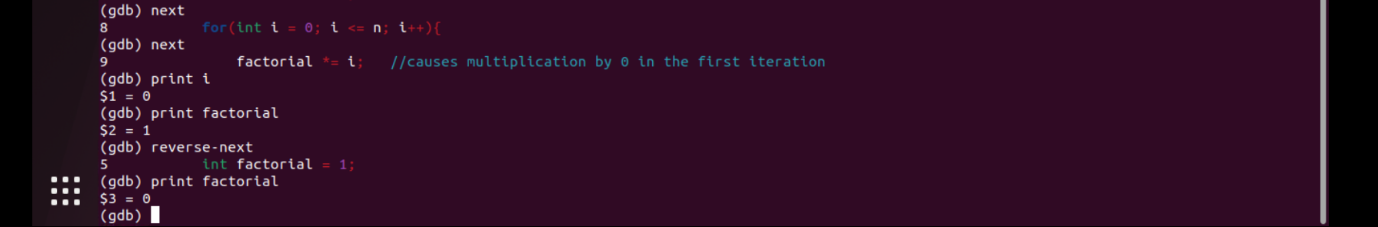
print factorial

reverse-next

print i

printfactorial





**Fix Logical Error:- fix\_logical\_error.c**

factorial\_debug\_fixed example.c

#include <stdio.h>

int main(){

    int n = 5;  //calculate factorial of 5

    int factorial = 1;

    //logical error: Incorrect initialization of the loop variable

    for(int i = 1; i <= n; i++){

        factorial \*= i;   //causes multiplication by 0 in the first iteration

    }

    printf("Factorial of %d is %d\n", n, factorial);

    return 0;

}

**Steps to debug with gdb to find logical error fixed or not :-**

gcc -g -o factorial\_debug\_fixed example.c

gdb ./factorial\_debug\_fixed

break main

target record-full

run

next

print i

print factorial

reverse-next

print i

print factorial

continue

run

