Module.4

4. Race Conditions in Multithreaded Code Description:

Reverse debugging is used to analyze nondeterministic bugs caused by race conditions.

Debugging Tasks:

1. Compile with threads

2. Use GDB:

Run the program and observe output.

Use record to enable reverse debugging.

Reverse-step through thread execution (reverse-next and reverse-step) to locate simultaneous access to counter.

3. Fix the issue using a mutex and retest.

**bug\_raceconditions\_multithread.c**

#include <stdio.h>

#include <pthread.h>

int shared\_counter = 0;  // Shared variable

void \*increment(void \*arg) {

    for (int i = 0; i < 10; i++) {

        shared\_counter++;  // Increment the shared variable

        printf("Incrementing: Counter = %d\n", shared\_counter);

    }

    return NULL;

}

void \*decrement(void \*arg) {

    for (int i = 0; i < 10; i++) {

        shared\_counter--;  // Decrement the shared variable

        printf("Decrementing: Counter = %d\n", shared\_counter);

    }

    return NULL;

}

int main() {

    pthread\_t thread1, thread2;

    // Create two threads: one for incrementing and one for decrementing

    pthread\_create(&thread1, NULL, increment, NULL);

    pthread\_create(&thread2, NULL, decrement, NULL);

    // Wait for both threads to finish

    pthread\_join(thread1, NULL);

    pthread\_join(thread2, NULL);

    printf("Final Counter Value: %d\n", shared\_counter);

    return 0;

}

**Step to find bug\_raceconditions\_multithread by debugging using gdb.**

Steps to Debug Race Condition

gcc -g -pthread -o bug\_raceconditions\_multithread bug\_raceconditions\_multithread.c

gdb ./bug\_raceconditions\_multithread

break increment

break decrement

run

record

next

print shared\_counter

reverse-next

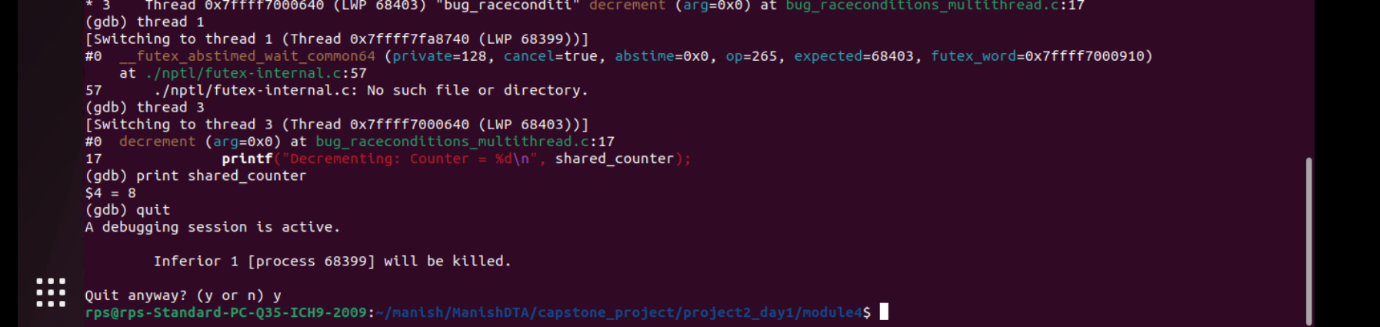
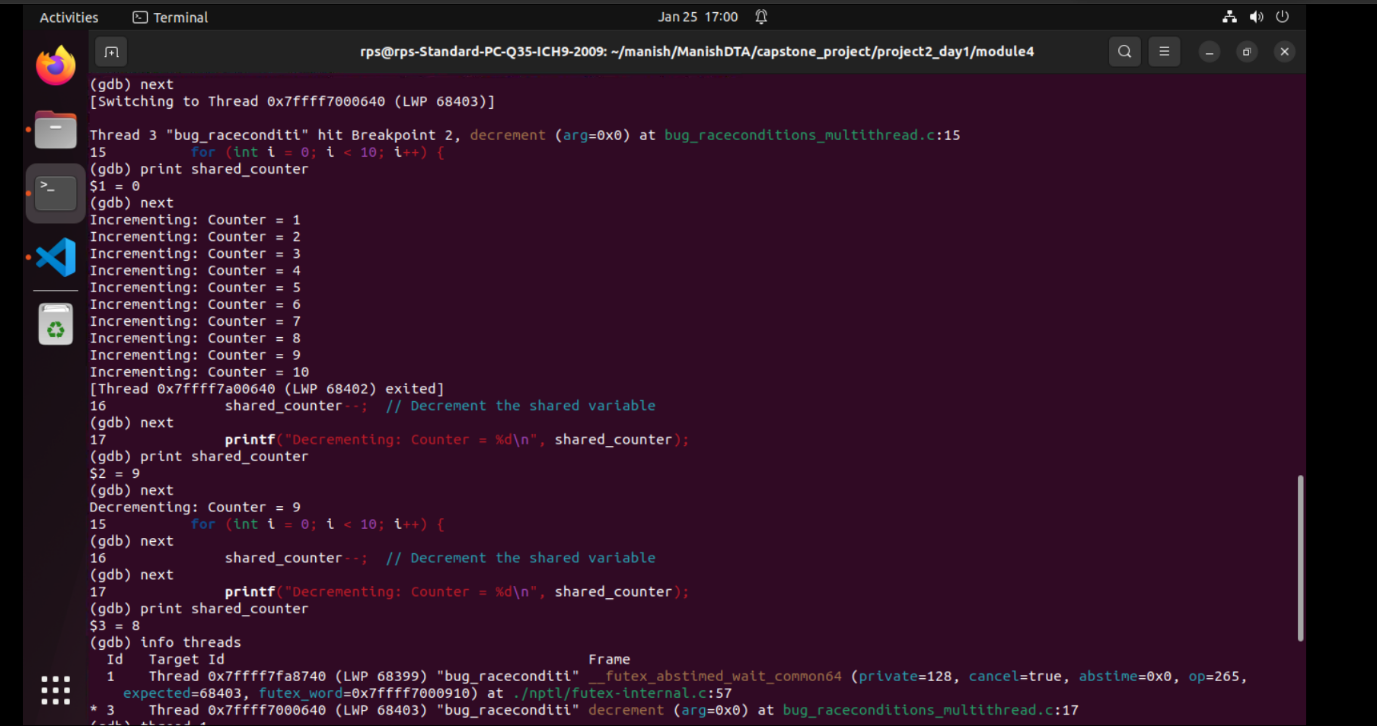
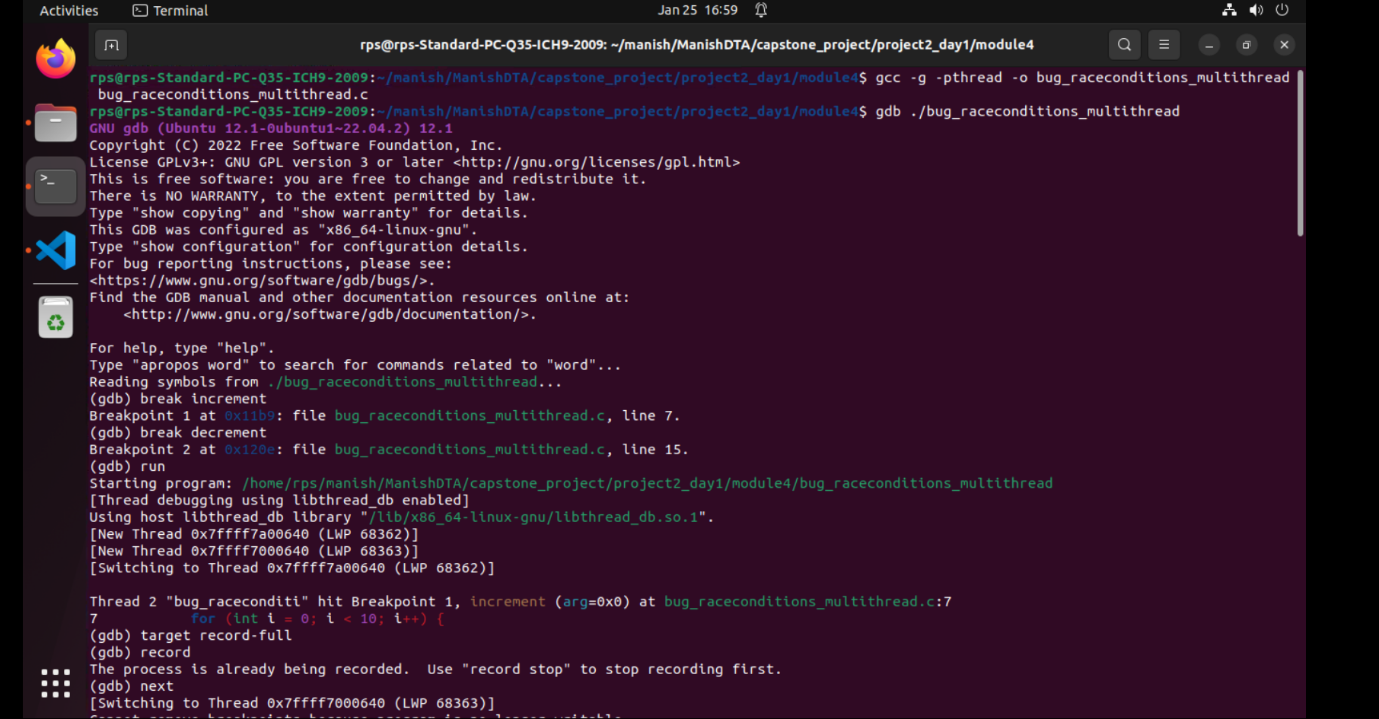
reverse-step

info threads

thread <thread-number>

forward or backward shared\_counter

delete record



**Fix the issue using a mutex support the multi threads and retest using gdb:-**

**fix\_bug\_raceconditions\_multithread.c**

#include <stdio.h>

#include <pthread.h>

int shared\_counter = 0;  // Shared variable

pthread\_mutex\_t lock;    // Mutex lock

void \*increment(void \*arg) {

    for (int i = 0; i < 10; i++) {

        pthread\_mutex\_lock(&lock);

        shared\_counter++;  // Safely increment

        printf("Incrementing: Counter = %d\n", shared\_counter);

        pthread\_mutex\_unlock(&lock);

    }

    return NULL;

}

void \*decrement(void \*arg) {

    for (int i = 0; i < 10; i++) {

        pthread\_mutex\_lock(&lock);

        shared\_counter--;  // Safely decrement

        printf("Decrementing: Counter = %d\n", shared\_counter);

        pthread\_mutex\_unlock(&lock);

    }

    return NULL;

}

int main() {

    pthread\_t thread1, thread2;

    pthread\_mutex\_init(&lock, NULL);  // Initialize the mutex

    // Create threads

    pthread\_create(&thread1, NULL, increment, NULL);

    pthread\_create(&thread2, NULL, decrement, NULL);

    // Wait for threads to finish

    pthread\_join(thread1, NULL);

    pthread\_join(thread2, NULL);

    pthread\_mutex\_destroy(&lock);  // Destroy the mutex

    printf("Final Counter Value: %d\n", shared\_counter);

    return 0;

}

**Steps to Verify if the Bug is Fixed or not using debugging with gdb**

gcc -g -pthread -o fix\_bug\_raceconditions\_multithread fix\_bug\_raceconditions\_multithread.c

gdb ./fix\_bug\_raceconditions\_multithread

break increment

break decrement

run

record

next

print shared\_counter

info threads

thread <thread-number>

reverse-next

reverse-step

delete record

