**EX 20:** **Construct a C program to simulate Reader-Writer problem using Semaphores.**

**Aim:**

To construct a C program that simulates the Reader-Writer Problem using semaphores.

**Algorithm:**

1. **Shared Resource**: A resource (like a variable) is shared by multiple readers and writers.
2. **Semaphores**:
   * mutex: Used for mutual exclusion between readers and writers.
   * write\_lock: Ensures that only one writer can access the resource at a time.
   * read\_count: Keeps track of the number of active readers.
3. **Reader**:
   * A reader can access the resource if no writer is currently writing.
   * Multiple readers can read the resource concurrently.
4. **Writer**:
   * A writer must wait until no reader is reading or another writer is writing.

**PROGRAM:**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>

#define MAX\_READERS 5

sem\_t mutex;

sem\_t wrt;

int read\_count = 0;

void\* reader(void\* arg) {

int f = \*((int\*)arg);

while (1) {

sem\_wait(&mutex);

read\_count++;

if (read\_count == 1) {

sem\_wait(&wrt);

}

sem\_post(&mutex);

printf("Reader %d is reading\n", f);

sleep(1);

sem\_wait(&mutex);

read\_count--;

if (read\_count == 0) {

sem\_post(&wrt);

}

sem\_post(&mutex);

sleep(1);

}

}

void\* writer(void\* arg) {

int f = \*((int\*)arg);

while (1) {

sem\_wait(&wrt);

printf("Writer %d is writing\n", f);

sleep(1);

sem\_post(&wrt);

sleep(1);

}

}

int main() {

pthread\_t read[MAX\_READERS], write;

int reader\_ids[MAX\_READERS];

sem\_init(&mutex, 0, 1);

sem\_init(&wrt, 0, 1);

for (int i = 0; i < MAX\_READERS; i++) {

reader\_ids[i] = i + 1;

pthread\_create(&read[i], NULL, reader, &reader\_ids[i]);

}

int writer\_id = 1;

pthread\_create(&write, NULL, writer, &writer\_id);

for (int i = 0; i < MAX\_READERS; i++) {

pthread\_join(read[i], NULL);

}

pthread\_join(write, NULL);

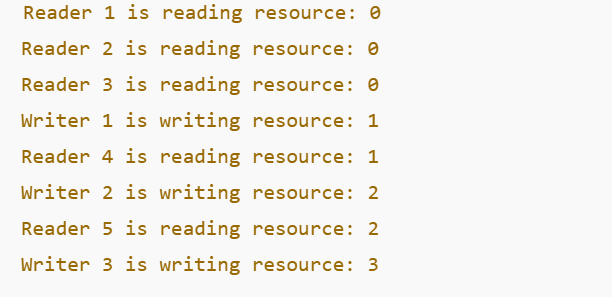
sem\_destroy(&mutex);

sem\_destroy(&wrt);

return 0;

}

**OUTPUT:**

****