**Swapping of Arrays using multithreading and managing it using semaphores in c**

#include *<stdio.h>*

#include *<stdlib.h>*

#include *<pthread.h>*

#include *<semaphore.h>*

#define SIZE 5

*// Global semaphore*

sem\_t semaphore;

*// Function to swap two arrays*

void\* swap\_arrays(void\* args) {

int\*\* arrays = (int\*\*)args; *// Cast to pointer to pointer to int*

int\* array1 = arrays[0];

int\* array2 = arrays[1];

*// Wait for semaphore*

sem\_wait(&semaphore);

*// Swap the arrays by reference*

**for** (int i = 0; i < SIZE; i++) {

int temp = array1[i];

array1[i] = array2[i];

array2[i] = temp;

}

*// Signal semaphore*

sem\_post(&semaphore);

**return** NULL;

}

int main() {

*// Example arrays*

int array1[] = {1,2,3,4,5};

int array2[] = {3,4,5,8,9};

*// Initialize the semaphore*

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

*// Prepare arguments for the thread*

int\*\* arrays = (int\*\*)malloc(2 \* **sizeof**(int\*)); *// Allocate for the array of pointers*

arrays[0] = array1;

arrays[1] = array2;

pthread\_t **thread**;

*// Create a thread to perform the swap*

pthread\_create(&**thread**, NULL, swap\_arrays, (void\*)arrays);

*// Wait for the thread to finish*

pthread\_join(**thread**, NULL);

*// Output the results*

printf("Array 1 after swap: ");

**for** (int i = 0; i < SIZE; i++) {

printf("%d ", arrays[0][i]); *// Access the swapped first array*

}

printf("**\n**");

printf("Array 2 after swap: ");

**for** (int i = 0; i < SIZE; i++) {

printf("%d ", arrays[1][i]); *// Access the swapped second array*

}

printf("**\n**");

*// Free allocated memory*

free(arrays);

*// Destroy the semaphore*

sem\_destroy(&semaphore);

**return** 0;

}

If the process uses multi threading to swap array elements then it becomes crucial to use semaphores as semaphores will allow us to manage and make sure that the data integrity remains,

if we didn’t use semaphores while using threads it would have ended up swapping random elements or worst swap the place that has already been swapped so to ensure that swapping takes place in the order that was intended we use semaphores to control it

Let us take the real world scenario where there are 2 classrooms and in 1 classroom say ff8 there are students of mca, and the other classroom say ff7 students of MSC. AI students are there and they want to exchange classrooms, if the students decide to change classrom randomly or without any order then there would be a congestion when students from ff8 are leaving and at the same time students from ff7 are entering ff8, this would result in a collision. So to orchestrate the process better we need a semaphore, lets say this semaphore is the class CR, hey would then swap students from ff8 to ff7 in a manner that no collisions occur and no 2 people are fighting to exit and enter at the same time. In this case without a semaphore it would cause chaos or even fights to enter and exit classrooms