EX.NO: 5 IMPLEMENTATION OF THREADING & SYNCHRONIZATION APPLICATIONS

Date:02.09.2024

AIM:

To write a C program to implement thread and synchronization application.

ALGORITHM:

Step1: Start the program.
Step2: i is initialized in the beginning of the main function.
Step3: Pthread function used to create thread by calling myThread().The variable is locked
using mutex variable i.
Step4: The threads wait for 30ms then the thread is completed one by one in the order which
will completed execution.
Step5: At the end of the main function the mutex is destroyed.
Step6: Stop the program.

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
void *myThreadFun(void *vargp)
{
   int thread_id = (int)(intptr_t)vargp;
   printf("Thread %d is processing\n", thread_id);
   for (volatile long i = 0; i < 100000000; i++);
   printf("Thread %d is completed\n", thread_id);
   pthread_exit(NULL);
}
int main() {
   int i;
   pthread_t a[5];
   printf("Before Thread\n");
   for (i = 0; i < 5; i++)
{
      if (pthread_create(&a[i], NULL, myThreadFun, (void *)(intptr_t)i) != 0)
{
         printf("Thread not created\n");
      }
   }
   for (i = 0; i < 5; i++) {
      pthread_join(a[i], NULL);
   }
   printf("All threads completed.\n");
   return 0;
}
```

OUTPUT:

```
Before Thread
Thread 0 is processing
Thread 1 is processing
Thread 2 is processing
Thread 3 is processing
Thread 4 is processing
Thread 0 is completed
Thread 1 is completed
Thread 2 is completed
Thread 3 is completed
Thread 4 is completed
All threads completed.
```

RESULT:

Thus, the program to implement thread synchronization using mutexes was executed successfully. Each thread processed its task without conflicts, ensuring safe access to the shared resource.