OPERATING SYSTEM LABORATORY MANUAL

UNIVERSITY OF THE PUNJAB

FACULTY OF COMPUTING & INFORMATION TECHNOLOGY, LAHORE

DEPARTMENT OF COMPUTER SCIENCE

Course:	**Operating System Lab**	**Date:**
Course Code:	**CC-217-3L**	**Max Marks: 40**
Faculty/Instructor's Name & Email:	**Dr. Ahmad Hassan Butt (ahmad.hassan@pucit.edu.pk)**	

LAB MANUAL # 11

(SPRING 2023)

Name:_____ Enroll No: _____

Objective(s) :

To understand Mutex and Thread Synchronization.

Lab Tasks :

Task 1 : Write the output of Hello World program using Threads.

Task 2: Write the outptut of program for Mutex Hello World.

Task 3 Identify the output error in Thread Synchronization Problem.

Task 4 : Write the output for Mutex for Thread Synchronization.

Lab Grading Sheet :

Task	Max Marks	Obtained Marks	Comments(*if any*)
1.	10		
2.	10		
3.	10		
4.	10		
Total	**40**		**Signature**

Note : Attempt all tasks and get them checked by your Instructor

Lab 11: Thread Synchronization

Objective(s):

To write a C program to understanding Mutex and Synchronization.

Tool(s) used:

Ubuntu, VIM Editor

Thread synchronization is defined as a mechanism which ensures that two or more concurrent processes or threads do not simultaneously execute some particular program segment known as critical section. Processes' access to critical section is controlled by using synchronization techniques. When one thread starts executing the critical section (serialized segment of the program) the other thread should wait until the first thread finishes. If proper synchronization techniques are not applied, it may cause a race condition where the values of variables may be unpredictable and vary depending on the timings of context switches of the processes or threads.

Task 01: Write the output of Hello World program using Threads.

```
#include <pthread.h>
#include <stdio.h>
void* compute_thread (void*);


int main( ){
	pthread_t tid;
	pthread_attr_t attr;
	char hello[ ] = {"Hello, "};
	char thread[ ] = {"thread"};
	pthread_attr_init(&attr);
	pthread_create(&tid, &attr, compute_thread, thread);
	printf(hello);
	sleep(1);
	printf("\n");
	exit(0);
	}
	void* compute_thread(void* dummy){
	printf(dummy);
	return 0;}
```

OUTPUT:

Task 02: Write the outptut of program for Mutex Hello World.

```
#include <pthread.h>
#include <stdio.h>
void* compute_thread (void*);
pthread_mutex_t my_sync;

main( ){
       pthread_t tid;
       pthread_attr_t attr;
       char hello[ ] = {"Hello, "};
       char thread[ ] = {"thread"};
       pthread_attr_init (&attr);
       pthread_mutex_init (&my_sync,NULL);
       pthread_create(&tid, &attr, compute_thread, hello);
       sleep(1);
       pthread_mutex_lock(&my_sync);
       printf(thread);
       printf("\n");
       pthread_mutex_unlock(&my_sync);
       exit(0);}

void* compute_thread(void* dummy){
       pthread_mutex_lock(&my_sync);
       printf(dummy);
       pthread_mutex_unlock(&my_sync);
       sleep(1);

return;
}
```

OUTPUT

Task 03: Identify the output error in Thread Synchronization Problem.

```
#include<stdio.h>
#include<string.h>
#include<pthread.h>
#include<stdlib.h>
#include<unistd.h>
pthread_t tid[2];
int counter;
void* trythis(void *arg){
        unsigned long i = 0;
        counter += 1;
        printf("\n Job %d has started\n", counter);
        for(i=0; i<(0xFFFFFFFF);i++);
        printf("\n Job %d has finished\n", counter);
        return NULL;
}

int main(void){
        int i = 0;
        int error;
        while(i < 2){
                error = pthread_create(&(tid[i]), NULL, &trythis, NULL);
                if (error != 0)
                        printf("\nThread can't be created : [%s]", strerror(error));
                i++;
        }
        pthread_join(tid[0], NULL);
        pthread_join(tid[1], NULL);
        return 0;
}
```

OUTPUT

Task 04: Write the output for Mutex for Thread Synchronization.

```
#include<stdio.h>
#include<string.h>
#include<pthread.h>
#include<stdlib.h>
#include<unistd.h>

pthread_t tid[2];
int counter;
pthread_mutex_t lock;
void* trythis(void *arg){
	pthread_mutex_lock(&lock);
	unsigned long i = 0;
	counter += 1;
	printf("\n Job %d has started\n", counter);
	for(i=0; i<(0xFFFFFFFF);i++);
	printf("\n Job %d has finished\n", counter);
	pthread_mutex_unlock(&lock);
	return NULL;
}

int main(void){
	int i = 0;
	int error;
	if (pthread_mutex_init(&lock, NULL) != 0){
		printf("\n mutex init has failed\n");
		return 1;
	}
	while(i < 2) {
		error = pthread_create(&(tid[i]), NULL, &trythis, NULL);
		if (error != 0)
			printf("\nThread can't be created :[%s]", strerror(error));
		i++;
	}
	pthread_join(tid[0], NULL);
	pthread_join(tid[1], NULL);
	pthread_mutex_destroy(&lock);
	return 0;
}
```

OUTPUT: