Jayant Gaur
20194152
CS-7B

DISTRIBUTED SYSTEMS LAB
ASSIGNMENT 2

Q1. Write a multithreaded program such that, there should be different threads for all different tasks and each thread access the file synchronously.

Solution:

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

pthread_t tid[3];
int counter;
pthread_mutex_t lock;

void* readFile(void* arg){
    pthread_mutex_lock(&lock);
    FILE *fptr=fopen("chk.txt","w");
        fputs("Hello, Mr. Pandey! ** ",fptr);
        fclose(fptr);
        printf("Write performed\n");
        pthread_mutex_unlock(&lock);
}

void* writeFile(void* arg){
    pthread_mutex_lock(&lock);
    char str[50];
        FILE *fptr=fopen("chk.txt","r");
        fgets(str,50,fptr);
        fclose(fptr);
        printf("Read performed\nStatement: %s",str);
        pthread_mutex_unlock(&lock);
}

void* updateFile(void* arg){
pthread_mutex_lock(&lock);
    char str[50];
        FILE *fptr=fopen("chk.txt","a");
        fputs("Hello, Abhijeet Pandey!\n",fptr);
        printf("Update performed\n");
        fclose(fptr);
        FILE *fptr1=fopen("chk.txt","r");
        fgets(str,50,fptr1);
        printf("Updated material: %s",str);
        fclose(fptr);
```

```
        pthread_mutex_unlock(&lock);
}

void* trythis(void* arg)
{
    int *idd = (int *)arg;
    pthread_mutex_lock(&lock);
    if(*idd==0)
    {

    }
    else if(*idd==1)
    {

    }
    else if(*idd==2)
    {

    }

    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;
    }

    error = pthread_create(&(tid[0]), NULL, &readFile, (void*)&i);

    error = pthread_create(&(tid[1]), NULL, &writeFile, (void*)&i);

    error = pthread_create(&(tid[2]), NULL, &updateFile, (void*)&i);
    if (error != 0)
            printf("\nThread can't be created :[%s]", strerror(error));
    while (i < 3)
    {
```

```
        i++;
    }

    pthread_join(tid[0], NULL);
    pthread_join(tid[1], NULL);
    pthread_join(tid[2], NULL);
    pthread_mutex_destroy(&lock);

    return 0;
}
```

Q2. Write a program to implement a deadlock scenario, in which two threads are accessing two resources concurrently.

Solution:

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

pthread_t tid[2];
pthread_mutex_t lock1, lock2;
int res_val = 0;

/* pthread1 runs here*/
void* func12()
{
    pthread_mutex_lock(&lock1);
    sleep(1);
    res_val += 15;
    printf("\nValue of res has been increased by 15, res_val = %d\n", res_val);
    pthread_mutex_lock(&lock2);
}

/* pthread2 runs here*/
void* func21()
{
    pthread_mutex_lock(&lock2);
    sleep(1);
    res_val += 39;
```

```
    printf("\nValue of res has been increased by 39, res_val = %d\n", res_val);
    pthread_mutex_lock(&lock1);
    printf("\nDeadlock demonstrated -- never gets printed");
}

int main()
{

    int i = 0;
    int error;
    if (pthread_mutex_init(&lock1, NULL) != 0 || pthread_mutex_init(&lock2, NULL)
!= 0)
    {
        printf("\n mutex init has failed\n");
        return 1;
    }

    while (i < 2)
    {
        if(i==0)
        {
            error = pthread_create(&(tid[i]), NULL, &func12, NULL);
            if (error != 0)
                printf("\nThread can't be created :[%s]", strerror(error));
        }
        else
        {
            error = pthread_create(&(tid[i]), NULL, &func21, 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(&lock1);
    pthread_mutex_destroy(&lock2);
    return 0;
}
```

Q3. Write a program to implement for deadlock avoidance using conditional locking in which two threads are accessing two resources concurrently. Note: userpthread_mutex_trylock() function for conational locking.

Solution:

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

pthread_t tid[2];
int counter;
pthread_mutex_t lock2;

/* pthread1 runs here*/
void *func1(void *param)
{  int done = 0;
   while (!done)
   {
      if (pthread_mutex_trylock(&lock2))
      {
         printf("\nWhat are you up to?");

         pthread_mutex_unlock(&lock2);
         done = 1;
      }
   }
   pthread_exit(0);
}

/* pthread2 runs here */
void *func2(void *param)
{  int done = 0;
   while (!done)
   {
     if (pthread_mutex_trylock(&lock2))
     {
         printf("\nHave a good day!");

         pthread_mutex_unlock(&lock2);
         done = 1;
     }
```

```
   }
   pthread_exit(0);
}

int main()
{

    int i = 0;
    int error;
    if (pthread_mutex_init(&lock2, NULL) != 0)
    {
        printf("\n mutex init has failed\n");
        return 1;
    }

    while (i < 2)
    {
        if(i==0)
        {
            error = pthread_create(&(tid[i]), NULL, &func1, NULL);
            if (error != 0)
                printf("\nThread can't be created :[%s]", strerror(error));
        }
        else
        {
            error = pthread_create(&(tid[i]), NULL, &func2, 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(&lock2);
    return 0;
}
```