Name : Ramzy Izza Wardhana
NIM : 21/472698/PA/20322
Class : IUP CS B

Assignment 7 – Lab Computer Networking and System

1. Create a program that sends a string "Hello Thread" to a thread that will store the string into a file titled "hello.txt".

Step 1: Create file inout_thread1.c using nano

```
root@--:~# nano inout_thread1.c
```

Step 2: Write the program in C

```
  GNU nano 6.2                          inout_thread1.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>

struct args {
    char* text;
};

void *createFile(void*content){
    int fileDesc;
    fileDesc = open("hello.txt", O_CREAT | O_WRONLY, S_IRUSR | S_IWUSR);
    write(fileDesc, content, 100);
    pthread_exit(NULL);
}

int main() {
    struct args *String = (struct args *)malloc(sizeof(struct args));
    char content[100] = "Hello Thread\n";
    String->text = content;

    pthread_t t;
    pthread_create(&t, NULL, createFile, content);
    pthread_join(t, NULL);
    return 0;
}
```

Step 3: Compile the program

```
root@--:~# gcc -o inout_thread1.out inout_thread1.c
```

Step 4: Run the Program

```
root@--:~# nano inout_thread1.c
root@--:~# gcc -o inout_thread1.out inout_thread1.c
root@--:~# ./inout_thread1.out
root@--:~# cat hello.txt
Hello Thread
root@--:~#
```

2. Create a program that simulates a deadlock using three threads.

Step 1 : Create file deadlock3.c using nano

```
root@--:~# nano deadlock3.c
```

Step 2: Write the code

Main Function:

```c
#include<stdio.h>
#include<pthread.h>
#include<unistd.h>
void *function1();
void *function2();
void *function3();
pthread_mutex_t resourceA;
pthread_mutex_t resourceB;
pthread_mutex_t resourceC;

int main() {

        pthread_mutex_init(&resourceA,NULL);
        pthread_mutex_init(&resourceB,NULL);
        pthread_mutex_init(&resourceC,NULL);

        pthread_t one, two, three;

        //create thread
        pthread_create(&one, NULL, function1, NULL);
        pthread_create(&two, NULL, function2, NULL);
        pthread_create(&three, NULL, function3, NULL);
        pthread_join(one, NULL);
        pthread_join(two, NULL);
        pthread_join(three, NULL);
        printf("Thread joined\n");
}
```

Function 1 & 2:

```c
void *function1() {
        pthread_mutex_lock(&resourceA);
        printf("Thread ONE acquired Resource A\n");
        sleep(1);

        pthread_mutex_lock(&resourceB);
        printf("Thread ONE acquired Resource B\n");

        pthread_mutex_unlock(&resourceB);
        printf("Thread ONE released Resource B\n");

        pthread_mutex_unlock(&resourceA);
        printf("Thread ONE released Resource A\n");
}


void *function2() {
        pthread_mutex_lock(&resourceB);
        printf("Thread TWO acquired Resource B\n");
        sleep(1);

        pthread_mutex_lock(&resourceC);
        printf("Thread TWO acquired Resource C\n");

        pthread_mutex_unlock(&resourceC);
        printf("Thread TWO released Resource C\n");

        pthread_mutex_unlock(&resourceB);
        printf("Thread TWO released Resource B\n");
}
```

Function 3:

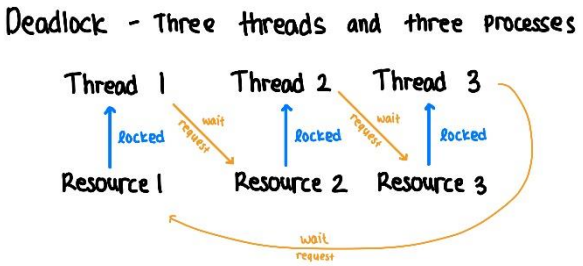
```c
void *function3() {
        pthread_mutex_lock(&resourceC);
        printf("Thread THREE acquired Resource C\n");
        sleep(1);

        pthread_mutex_lock(&resourceA);
        printf("Thread THREE acquired Resource A\n");

        pthread_mutex_unlock(&resourceA);
        printf("Thread THREE released Resource A\n");

        pthread_mutex_unlock(&resourceC);
        printf("Thread THREE released Resource C\n");
}
```

Deadlock Diagram:




Step 3: Compile the program

```
root@--:~# nano deadlock3.c
root@--:~# gcc -o deadlock3.out deadlock3.c
```

Step 4: Run the Program

```
root@--:~# ./deadlock3.out
Thread ONE acquired Resource A
Thread THREE acquired Resource C
Thread TWO acquired Resource B
```

Observe that thread one, two, three has acquired each resource but they have to wait indefinitely which indicated as a deadlock.