



[Workshop Report-3]

Student Id : NP03A190051
Student Name : Rishab Sharma
Section : L5CG7
Lecturer : Jnaneshwor Bohara
Module Leader : Hiran Patel
Submitted on : 1st October, 2021

1. 1. The following program demonstrates 3 thread sending string messages to each other, using a global array. The messages are sent meant to be sent in the following order:
 - a. Thread 0 sends Thread 1 a message
 - b. Thread 1 receives the message
 - c. Thread 1 sends Thread 2 a message
 - d. Thread 2 receives the message
 - e. Thread 2 sends Thread 0 a message
 - f. Thread 0 receives the message
 - g. This then repeats from (a) 10 times

```
=> #include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <string.h>
```

```
#include <pthread.h>
```

```
#include <unistd.h>
```

```
char *messages[3] = {NULL, NULL, NULL};
```

```
void *messenger(void *p)
```

```
{
```

```
    long tid = (long)p;
```

```
    char tmpbuf[100];
```

```
    for(int i=0; i<10; i++)
```

```
{
```

```
    /* Sending a message */
```

```
    long int dest = (tid + 1) % 3;
```

```
    sprintf(tmpbuf, "Hello from Thread %ld!", tid);
```

```
    char *msg = strdup(tmpbuf);
```

```
    messages[dest] = msg;
```

```
    printf("Thread %ld sent the message to Thread %ld\n", tid, dest);
```

```
    /* Receiving a message */
```

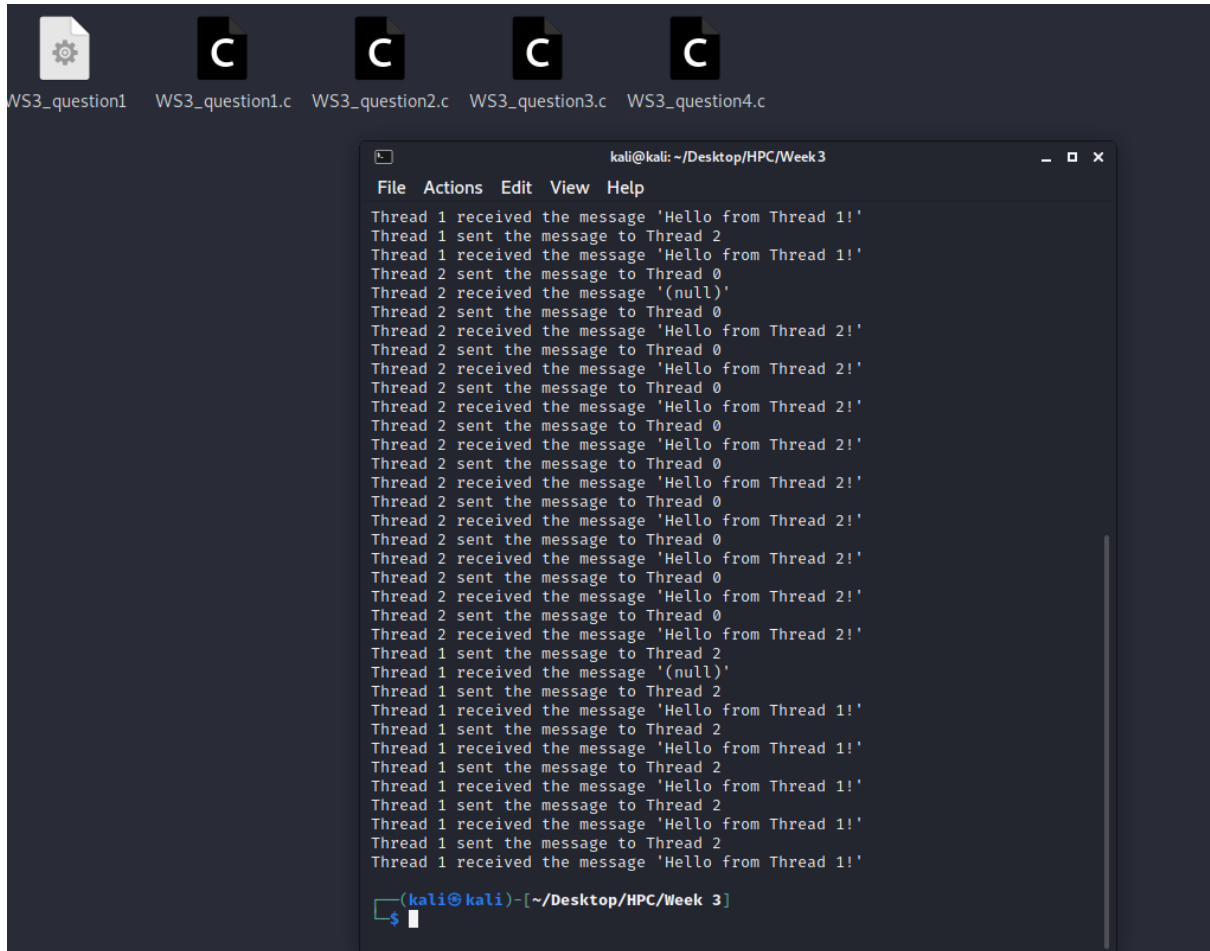
```
    printf("Thread %ld received the message '%s'\n", tid, messages[dest]);
```

```
    free(messages[tid]);
```

```
    messages[tid] = NULL;
```

```
void main() {  
    pthread_t thrID1, thrID2, thrID3;  
    pthread_create(&thrID1, NULL, messenger, (void *)0);  
    pthread_create(&thrID2, NULL, messenger, (void *)1);  
    pthread_create(&thrID3, NULL, messenger, (void *)2);  
    pthread_join(thrID1, NULL);  
    pthread_join(thrID2, NULL);  
    pthread_join(thrID3, NULL);  
}
```

[illegible]



- Using the technique of “busy-waiting” to correct the program, and establishing the correct order of messages.

```
=> #include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <string.h>
```

```
#include <pthread.h>
```

```
#include <unistd.h>
```

```
char *messages[3] = {NULL, NULL, NULL};
```

```
int flag = 0;
```

```
void *messenger(void *p)
```

{

```
long tid = (long)p;
```

```

char tmpbuf[100];
    for(int i=0; i<10; i++)
    {
        while(flag!=tid);
        /* Sending a message */
        long int dest = (tid + 1) % 3;
        sprintf(tmpbuf,"Hello from Thread %ld!", tid);
        char *msg = strdup(tmpbuf);
        messages[dest] = msg;
        printf("Thread %ld sent the message to Thread %ld\n",tid, dest);

        /* Receiving a message */
        printf("Thread %ld received the message '%s'\n",dest, messages[dest]);
        free(messages[dest]);
        messages[dest] = NULL;
        flag = dest;
    }
    return NULL;
}

```

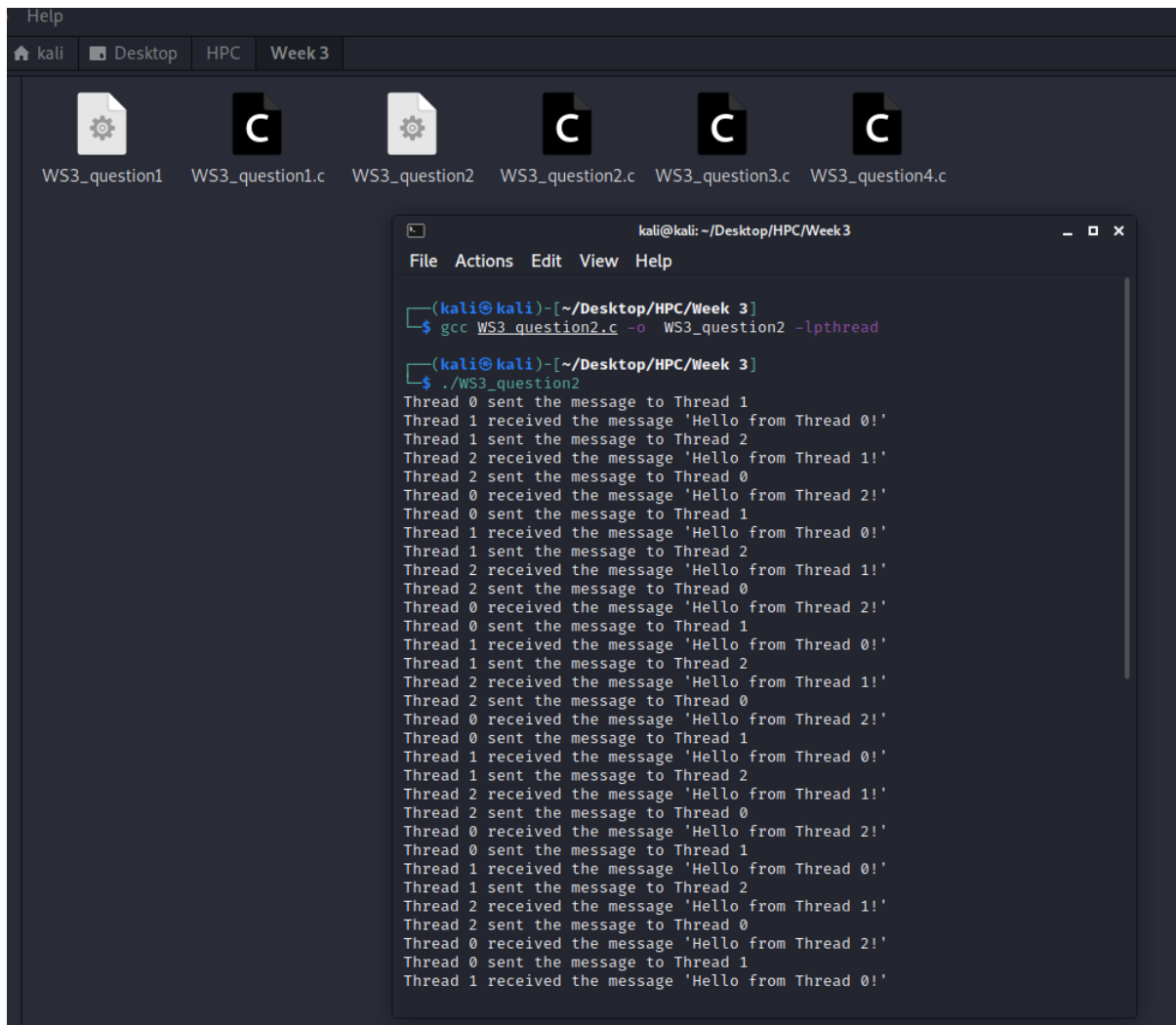
```

void main()
{
    pthread_t thrID1, thrID2, thrID3;
    pthread_create(&thrID1, NULL, messenger, (void *)0);
    pthread_create(&thrID2, NULL, messenger, (void *)1);
    pthread_create(&thrID3, NULL, messenger, (void *)2);
    pthread_join(thrID1, NULL);
    pthread_join(thrID2, NULL);
}

```

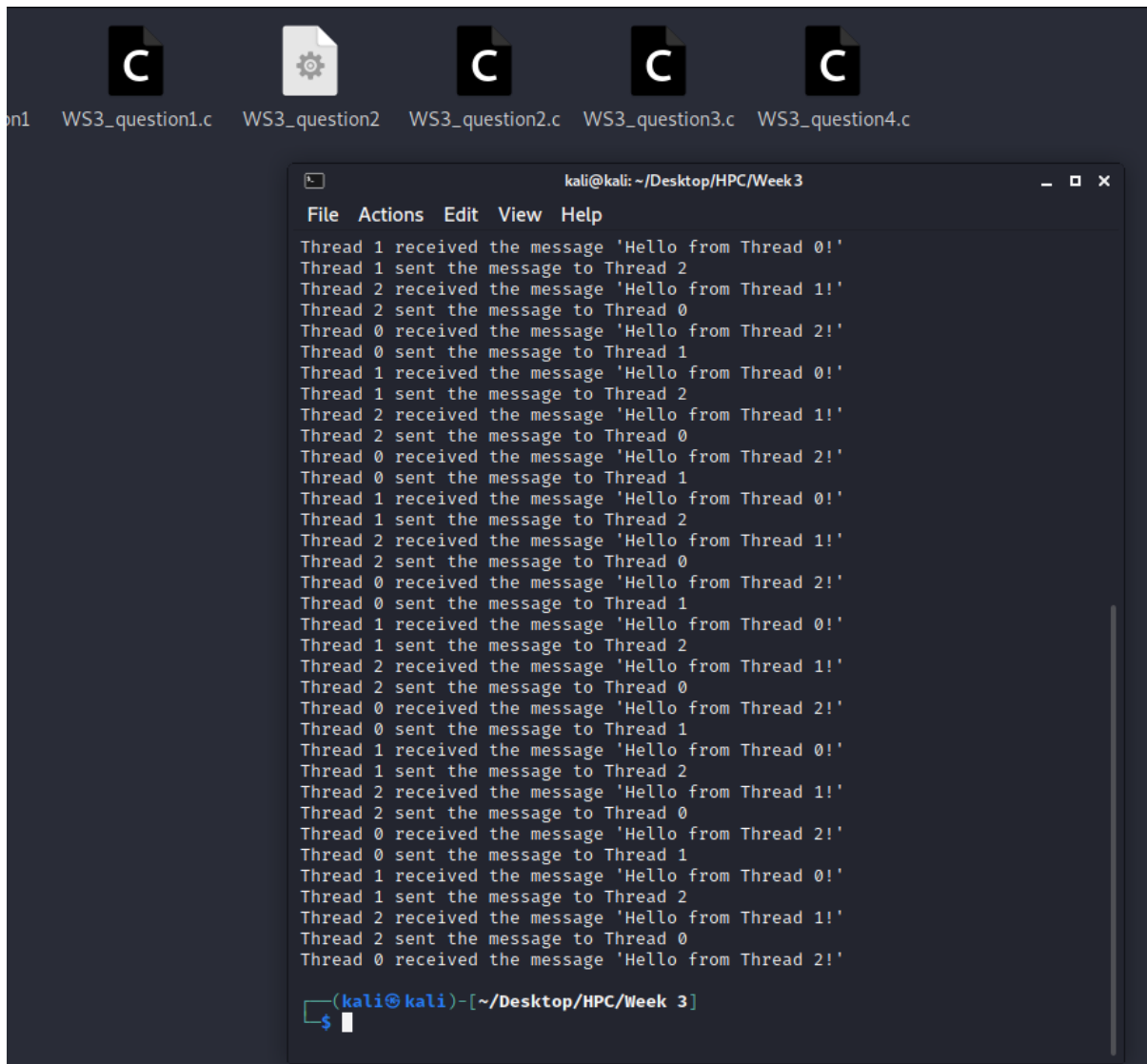
```
pthread_join(thrID3, NULL);  
}
```

OUTPUT: -



The screenshot shows a Kali Linux desktop environment. At the top, there is a taskbar with icons for 'kali', 'Desktop', 'HPC', and 'Week 3'. Below the taskbar, a file manager window displays a directory containing several files: 'WS3_question1', 'WS3_question1.c', 'WS3_question2', 'WS3_question2.c', 'WS3_question3.c', and 'WS3_question4.c'. In the foreground, a terminal window is open, showing the following commands and output:

```
kali@kali: ~/Desktop/HPC/Week 3  
File Actions Edit View Help  
  
(kali@kali)~[~/Desktop/HPC/Week 3]  
$ gcc WS3_question2.c -o WS3_question2 -lpthread  
  
(kali@kali)~[~/Desktop/HPC/Week 3]  
$ ./WS3_question2  
Thread 0 sent the message to Thread 1  
Thread 1 received the message 'Hello from Thread 0!'  
Thread 1 sent the message to Thread 2  
Thread 2 received the message 'Hello from Thread 1!'  
Thread 2 sent the message to Thread 0  
Thread 0 received the message 'Hello from Thread 2!'  
Thread 0 sent the message to Thread 1  
Thread 1 received the message 'Hello from Thread 0!'  
Thread 1 sent the message to Thread 2  
Thread 2 received the message 'Hello from Thread 1!'  
Thread 2 sent the message to Thread 0  
Thread 0 received the message 'Hello from Thread 2!'  
Thread 0 sent the message to Thread 1  
Thread 1 received the message 'Hello from Thread 0!'  
Thread 1 sent the message to Thread 2  
Thread 2 received the message 'Hello from Thread 1!'  
Thread 2 sent the message to Thread 0  
Thread 0 received the message 'Hello from Thread 2!'  
Thread 0 sent the message to Thread 1  
Thread 1 received the message 'Hello from Thread 0!'  
Thread 1 sent the message to Thread 2  
Thread 2 received the message 'Hello from Thread 1!'  
Thread 2 sent the message to Thread 0  
Thread 0 received the message 'Hello from Thread 2!'  
Thread 0 sent the message to Thread 1  
Thread 1 received the message 'Hello from Thread 0!'  
Thread 1 sent the message to Thread 2  
Thread 2 received the message 'Hello from Thread 1!'  
Thread 2 sent the message to Thread 0  
Thread 0 received the message 'Hello from Thread 2!'  
Thread 0 sent the message to Thread 1  
Thread 1 received the message 'Hello from Thread 0!'
```



The screenshot shows a Kali Linux desktop with a dark theme. In the background, there are five file icons labeled 'WS3_question1.c', 'WS3_question2', 'WS3_question2.c', 'WS3_question3.c', and 'WS3_question4.c'. In the foreground, a terminal window is open with the title 'kali@kali: ~/Desktop/HPC/Week3'. The terminal displays the output of a program with three threads (Thread 0, Thread 1, and Thread 2) that repeatedly send and receive the message 'Hello from Thread X!'. The output shows a sequence of messages where each thread sends a message to the others in a repeating cycle. At the bottom of the terminal, the prompt '(kali@kali)~[~/Desktop/HPC/Week 3]' is visible, followed by a '\$' symbol and a cursor.

```
kali@kali: ~/Desktop/HPC/Week3
File Actions Edit View Help
Thread 1 received the message 'Hello from Thread 0!'
Thread 1 sent the message to Thread 2
Thread 2 received the message 'Hello from Thread 1!'
Thread 2 sent the message to Thread 0
Thread 0 received the message 'Hello from Thread 2!'
Thread 0 sent the message to Thread 1
Thread 1 received the message 'Hello from Thread 0!'
Thread 1 sent the message to Thread 2
Thread 2 received the message 'Hello from Thread 1!'
Thread 2 sent the message to Thread 0
Thread 0 received the message 'Hello from Thread 2!'
Thread 0 sent the message to Thread 1
Thread 1 received the message 'Hello from Thread 0!'
Thread 1 sent the message to Thread 2
Thread 2 received the message 'Hello from Thread 1!'
Thread 2 sent the message to Thread 0
Thread 0 received the message 'Hello from Thread 2!'
Thread 0 sent the message to Thread 1
Thread 1 received the message 'Hello from Thread 0!'
Thread 1 sent the message to Thread 2
Thread 2 received the message 'Hello from Thread 1!'
Thread 2 sent the message to Thread 0
Thread 0 received the message 'Hello from Thread 2!'
Thread 0 sent the message to Thread 1
Thread 1 received the message 'Hello from Thread 0!'
Thread 1 sent the message to Thread 2
Thread 2 received the message 'Hello from Thread 1!'
Thread 2 sent the message to Thread 0
Thread 0 received the message 'Hello from Thread 2!'
Thread 0 sent the message to Thread 1
Thread 1 received the message 'Hello from Thread 0!'
Thread 1 sent the message to Thread 2
Thread 2 received the message 'Hello from Thread 1!'
Thread 2 sent the message to Thread 0
Thread 0 received the message 'Hello from Thread 2!'

(kali@kali)~[~/Desktop/HPC/Week 3]
$
```

3. Use pthread “mutex” to correct the program in (1). You will need multiple mutexes.

```
=> #include <stdio.h>
```

```
    #include <stdlib.h>
```

```
    #include <string.h>
```

```
    #include <pthread.h>
```

```
    #include <unistd.h>
```

```
char *messages[3] = {NULL, NULL, NULL};
```

```
int flag = 0;
```

```
pthread_mutex_t mutex;
```

```

void *messenger(void *p)
{
    long tid = (long)p;
    char tmpbuf[100];
    for(int i=0; i<10; i++)
    {
        pthread_mutex_lock(&mutex);

        /* Sending a message */

        long int dest = (tid + 1) % 3;
        sprintf(tmpbuf,"Hello from Thread %ld!", tid);
        char *msg = strdup(tmpbuf);
        messages[dest] = msg;
        printf("Thread %ld sent the message to Thread %ld\n",tid, dest);

        /* Receiving a message */

        printf("Thread %ld received the message '%s'\n",dest, messages[dest]);
        free(messages[dest]);
        messages[dest] = NULL;
        pthread_mutex_unlock(&mutex);
    }
    return NULL;
}

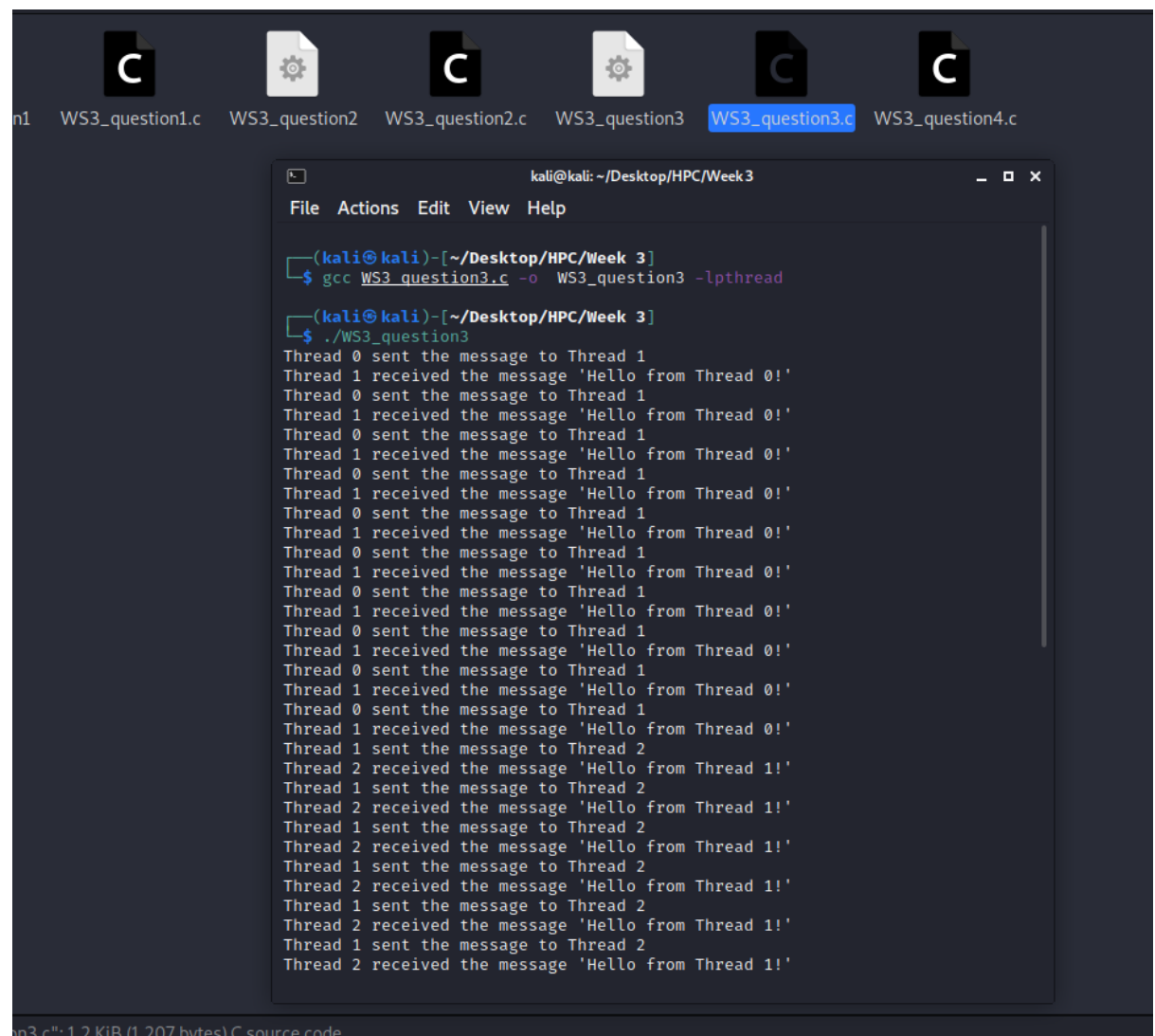
```

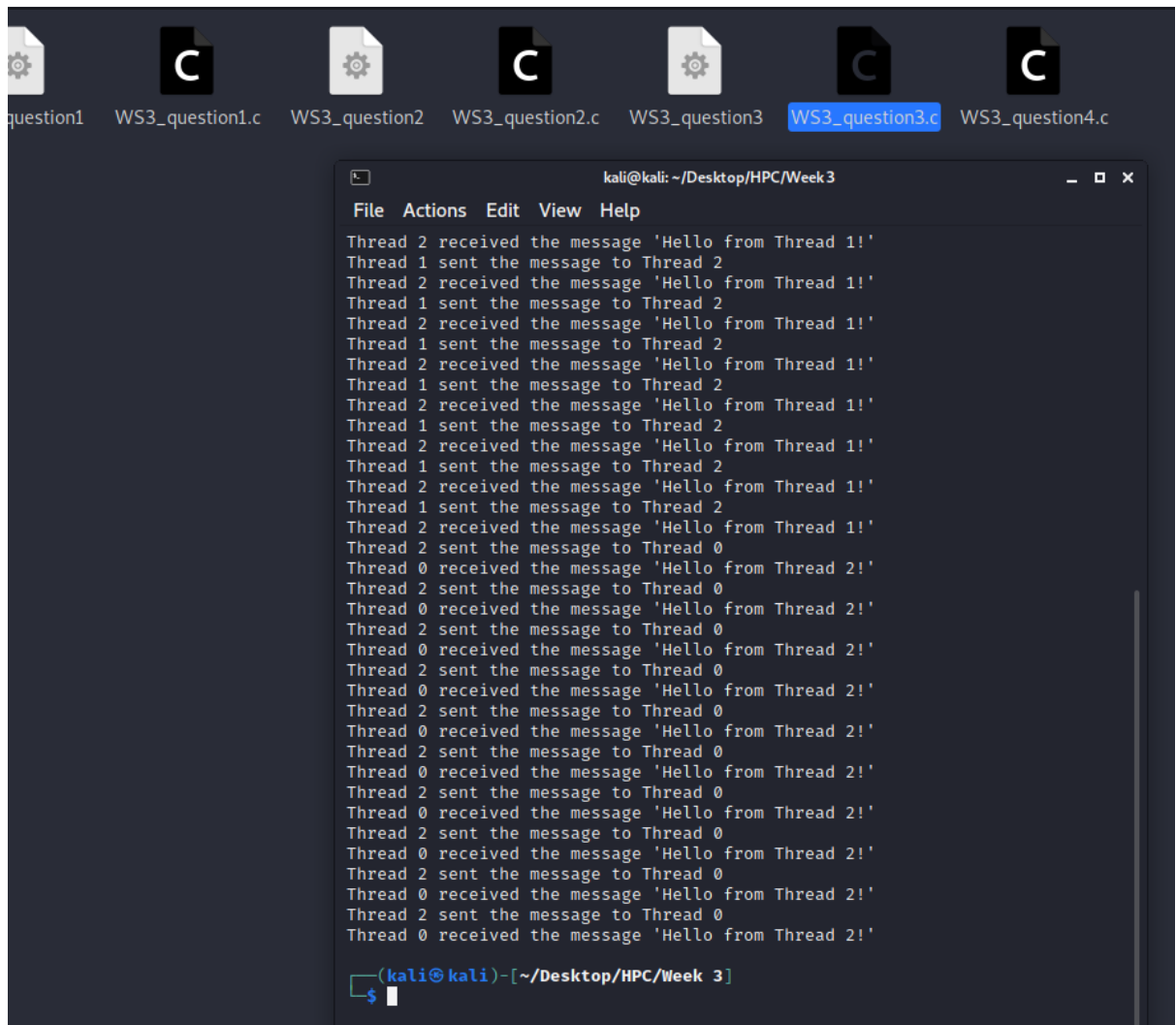
```

void main()
{
    pthread_t thrID1, thrID2, thrID3;

```


OUTPUT: -



A screenshot of a Kali Linux desktop environment. The top panel shows several file icons, including 'question1', 'WS3_question1.c', 'WS3_question2', 'WS3_question2.c', 'WS3_question3', 'WS3_question3.c' (highlighted in blue), and 'WS3_question4.c'. Below the panel is a terminal window titled 'kali@kali: ~/Desktop/HPC/Week3'. The terminal displays a series of messages between three threads (Thread 0, Thread 1, and Thread 2). The messages show Thread 1 sending 'Hello from Thread 1!' to Thread 2, Thread 2 sending 'Hello from Thread 2!' to Thread 0, and Thread 0 sending 'Hello from Thread 0!' to Thread 1. The messages are repeated multiple times, indicating a loop of communication. The terminal prompt is '\$'.

4. Use semaphores to correct the program in (1).

=> #include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <pthread.h>

#include <unistd.h>

#include<semaphore.h>

char *messages[3] = {NULL, NULL, NULL};

int flag = 0;

sem_t lock;

void *messenger(void *p)

```

{
    long tid = (long)p;
    char tmpbuf[100];
    for(int i=0; i<10; i++)
    {
        sem_wait(&lock);
        /* Sending a message */
        long int dest = (tid + 1) % 3;
        sprintf(tmpbuf,"Hello from Thread %d!", tid);
        char *msg = strdup(tmpbuf);
        messages[dest] = msg;
        printf("Thread %d sent the message to Thread %d\n",tid, dest);

        /* Receiving a message */

        printf("Thread %d received the message '%s'\n",dest, messages[dest]);
        free(messages[dest]);
        messages[dest] = NULL;
        sem_post(&lock);
    }
    return NULL;
}

void main()
{
    pthread_t thrID1, thrID2, thrID3;
    sem_init(&lock, 1, 1);
    pthread_create(&thrID1, NULL, messenger, (void *)0);
    pthread_create(&thrID2, NULL, messenger, (void *)1);
    pthread_create(&thrID3, NULL, messenger, (void *)2);

```

```

pthread_join(thrID1, NULL);
pthread_join(thrID2, NULL);
pthread_join(thrID3, NULL);
sem_destroy(&lock);
}

```

OUTPUT: -

```

kali@kali: ~/Desktop/HPC/Week 3
File Actions Edit View Help

(kali@kali)-[~/Desktop/HPC/Week 3]
$ gcc WS3_question4.c -o WS3_question4 -lpthread

(kali@kali)-[~/Desktop/HPC/Week 3]
$ ./WS3_question4
Thread 0 sent the message to Thread 1
Thread 1 received the message 'Hello from Thread 0!'
Thread 0 sent the message to Thread 1
Thread 1 received the message 'Hello from Thread 0!'
Thread 0 sent the message to Thread 1
Thread 1 received the message 'Hello from Thread 0!'
Thread 0 sent the message to Thread 1
Thread 1 received the message 'Hello from Thread 0!'
Thread 0 sent the message to Thread 1
Thread 1 received the message 'Hello from Thread 0!'
Thread 0 sent the message to Thread 1
Thread 1 sent the message to Thread 2
Thread 2 received the message 'Hello from Thread 1!'
Thread 1 sent the message to Thread 2
Thread 2 received the message 'Hello from Thread 1!'
Thread 1 sent the message to Thread 2
Thread 2 received the message 'Hello from Thread 1!'
Thread 0 sent the message to Thread 1
Thread 1 received the message 'Hello from Thread 0!'
Thread 0 sent the message to Thread 1
Thread 1 received the message 'Hello from Thread 0!'
Thread 0 sent the message to Thread 1
Thread 1 received the message 'Hello from Thread 0!'
Thread 1 sent the message to Thread 2
Thread 2 received the message 'Hello from Thread 1!'
Thread 1 sent the message to Thread 2
Thread 2 received the message 'Hello from Thread 1!'

```

WS3.c": 1.2 KiB (1.207 bytes) C source code



WS3_question4.c

— □ ×

```
Thread 1 received the message 'Hello from Thread 0!'
Thread 0 sent the message to Thread 1
Thread 1 received the message 'Hello from Thread 0!'
Thread 1 sent the message to Thread 2
Thread 2 received the message 'Hello from Thread 1!'
Thread 1 sent the message to Thread 2
Thread 2 received the message 'Hello from Thread 1!'
Thread 1 sent the message to Thread 2
Thread 2 received the message 'Hello from Thread 1!'
Thread 1 sent the message to Thread 2
Thread 2 received the message 'Hello from Thread 1!'
Thread 2 sent the message to Thread 0
Thread 0 received the message 'Hello from Thread 2!'
Thread 2 sent the message to Thread 0
Thread 0 received the message 'Hello from Thread 2!'
Thread 2 sent the message to Thread 0
Thread 0 received the message 'Hello from Thread 2!'
Thread 2 sent the message to Thread 0
Thread 0 received the message 'Hello from Thread 2!'
Thread 2 sent the message to Thread 0
Thread 0 received the message 'Hello from Thread 2!'
Thread 2 sent the message to Thread 0
Thread 0 received the message 'Hello from Thread 2!'
Thread 2 sent the message to Thread 0
Thread 0 received the message 'Hello from Thread 2!'
Thread 2 sent the message to Thread 0
Thread 0 received the message 'Hello from Thread 2!'
Thread 2 sent the message to Thread 0
Thread 0 received the message 'Hello from Thread 2!'
Thread 1 sent the message to Thread 2
Thread 2 received the message 'Hello from Thread 1!'
Thread 1 sent the message to Thread 2
Thread 2 received the message 'Hello from Thread 1!'
```

```
(kali㉿kali)-[~/Desktop/HPC/Week 3]
```