

Student Id : NP03A190051

Student Name : Rishab Sharma

Section : L5CG7

Lecturer : Jnaneshwor Bohara

Module Leader : Hiran Patel

Submitted on : 1st October, 2021

[Workshop Report-3]

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);

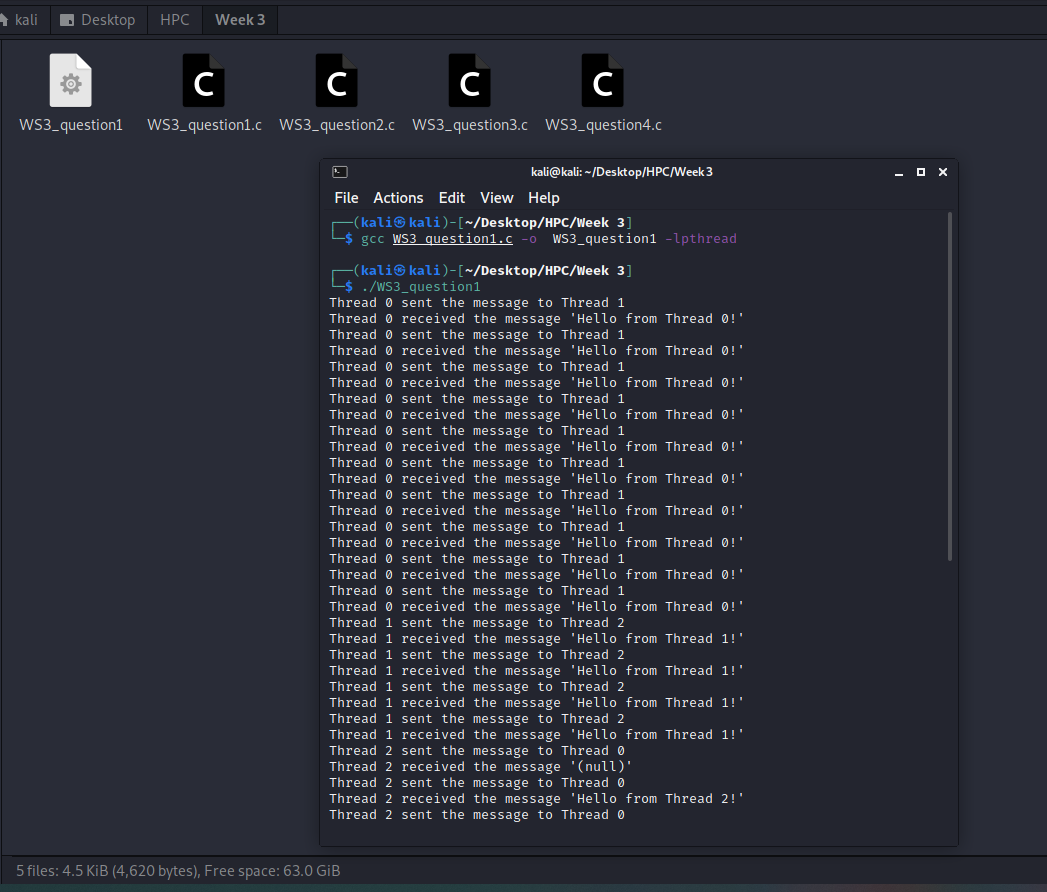
pthread\_join(thrID1, NULL);

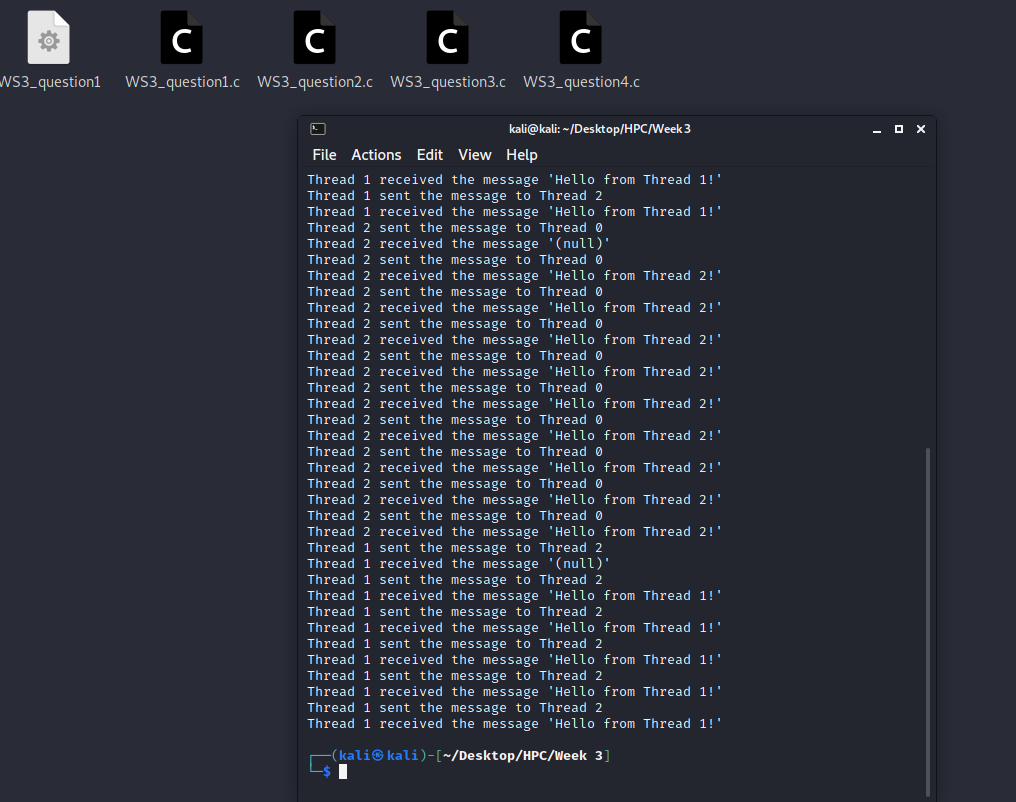
pthread\_join(thrID2, NULL);

pthread\_join(thrID3, NULL);

}

OUTPUT: -





1. 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);

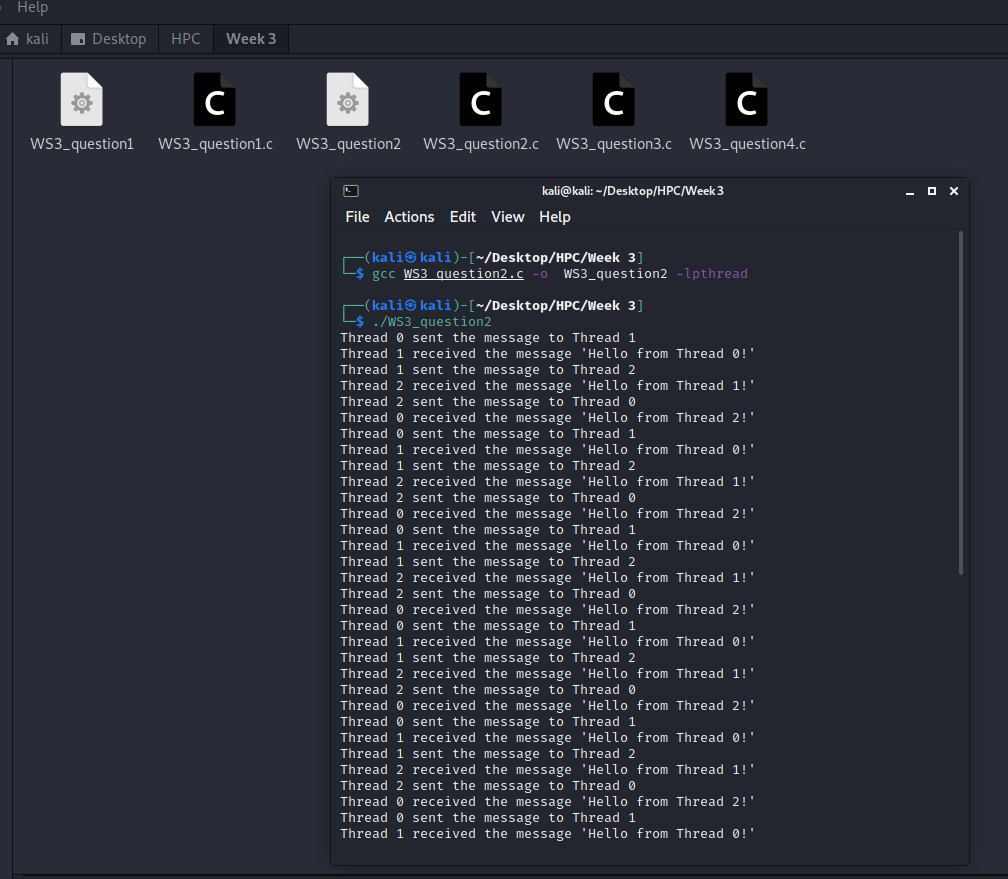
pthread\_join(thrID1, NULL);

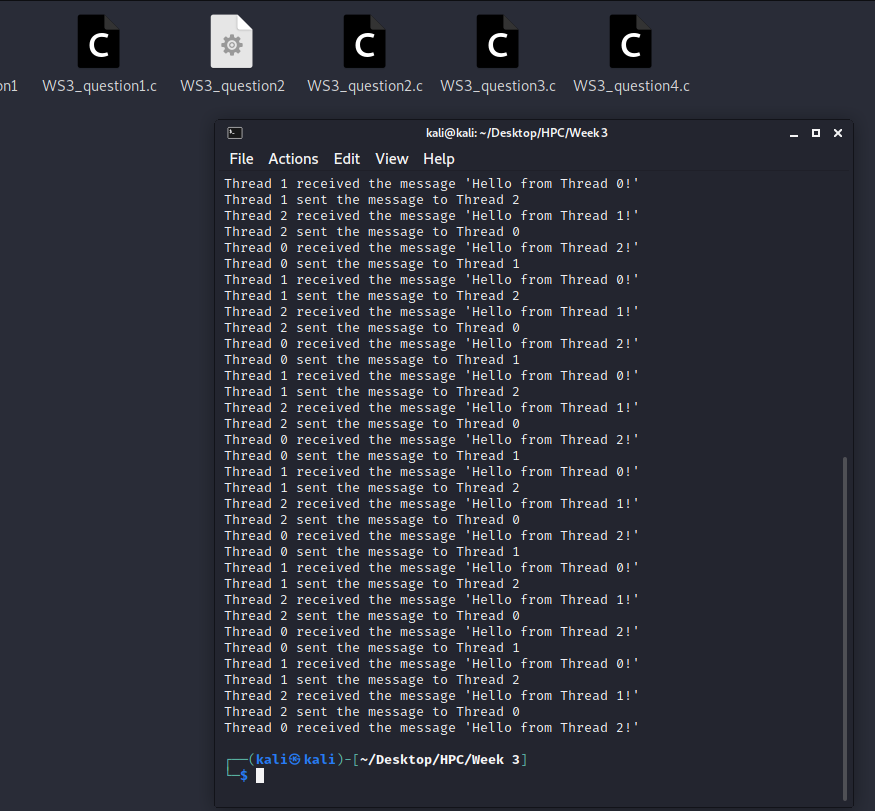
pthread\_join(thrID2, NULL);

pthread\_join(thrID3, NULL);

}

OUTPUT: -





1. 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;

pthread\_create(&thrID1, NULL, messenger, (void \*)0);

pthread\_create(&thrID2, NULL, messenger, (void \*)1);

pthread\_create(&thrID3, NULL, messenger, (void \*)2);

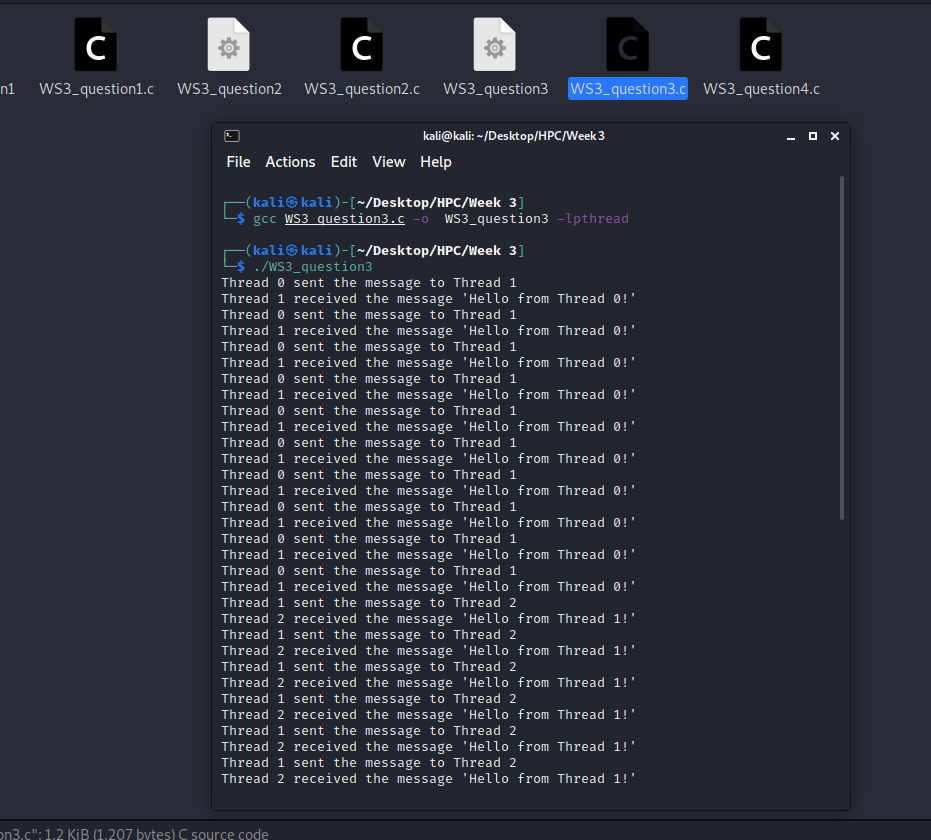
pthread\_join(thrID1, NULL);

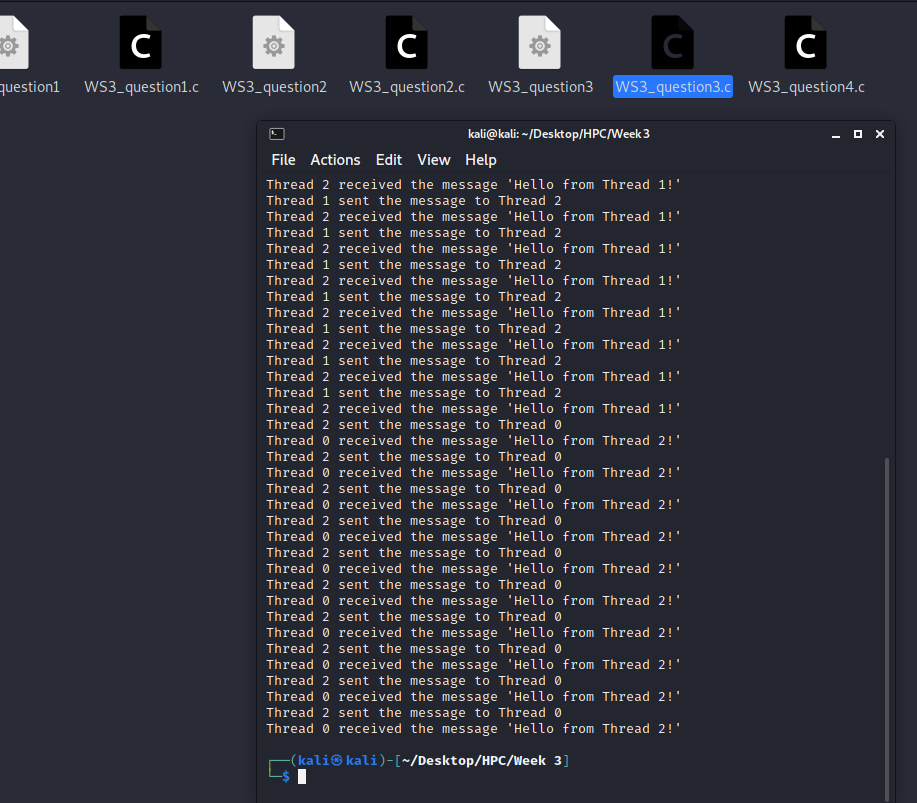
pthread\_join(thrID2, NULL);

pthread\_join(thrID3, NULL);

}

OUTPUT: -





1. 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 %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;

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: -

