Systems Programming Assignment # 3

Name: Syed Muhammad Ashhar Shah
Reg No: 2020478

Question 1

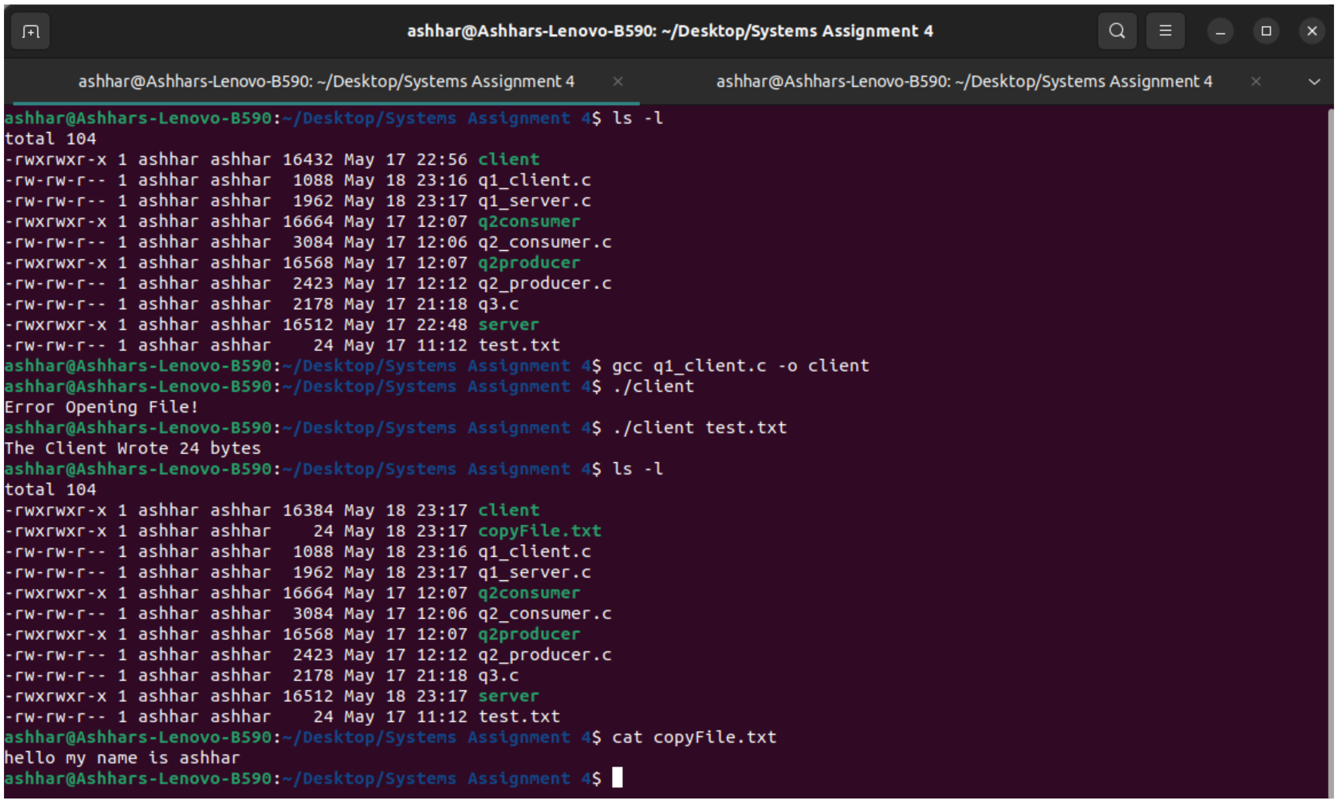
Client

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

int main(int argc, char* argv[]){
// variables for the socket
int sockfd;
struct sockaddr_in address;
int result;
char toServ;
int bytesWritten = 0;

// get the name of the file
char* fileName = argv[1];
// open the file
int fd = open(fileName, O_RDONLY);
if(fd == -1){
printf("Error Opening File!\n");
exit(1);
}

sockfd = socket(AF_INET, SOCK_STREAM, 0);
address.sin_family = AF_INET;
address.sin_addr.s_addr = htonl(INADDR_ANY);
address.sin_port = htons(9734);

result = connect(sockfd, (struct sockaddr *) &address, sizeof(address));
if(result == -1){
printf("Error In client!\n");
exit(1);
}

while(read(fd, &toServ, 1) == 1){
write(sockfd, &toServ, 1);
bytesWritten++;
}

printf("The Client Wrote %d bytes\n", bytesWritten);

close(sockfd);
exit(0);

return 0;
}
```

Server

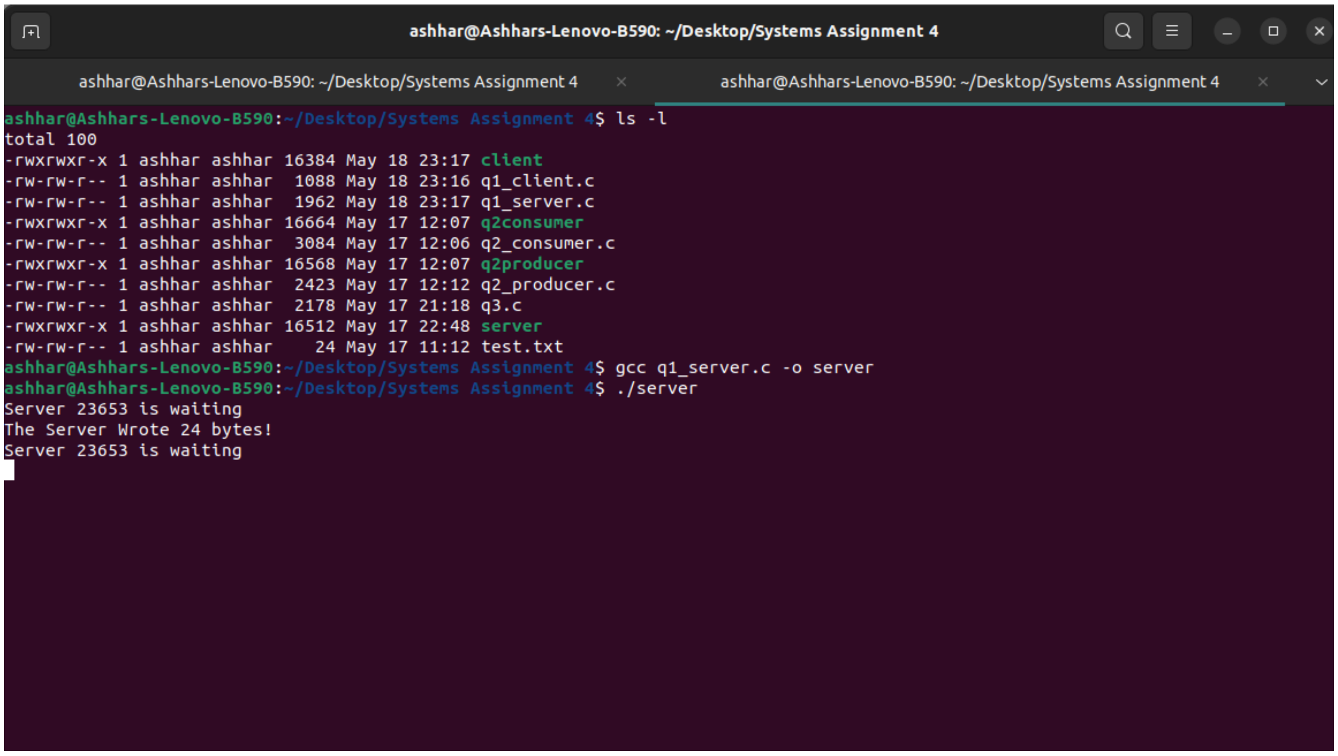
```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

int main(){
// file descriptors for the client and server sockets
int S_sockfd;
int C_sockfd;

// used to store legnths of the server and client socket address
int S_len;
int C_len;
// variables to store the client and server addresses
struct sockaddr_in S_addr;
struct sockaddr_in C_addr;

// variable to store incomming message
char fromClient;

// server configuration
S_addr.sin_family = AF_INET;
S_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
S_addr.sin_port = htons(9734);
S_len = sizeof(S_addr);

//creation of a socket
S_sockfd = socket(AF_INET, SOCK_STREAM, 0);

// bind the socket to the address/ports
bind(S_sockfd, (struct sockaddr *) &S_addr, S_len);

// make socket listen for incomming connections
listen(S_sockfd, 5);


while(1){
// print that server is waiting for an incomming connection
printf("Server %d is waiting\n", getpid());

// configure client fd on a connection accept
C_len = sizeof(C_addr);
C_sockfd = accept(S_sockfd, (struct sockaddr *) &C_addr, &C_len);

// open the file to which we are supposed to write to
int fd = open("copyFile.txt", O_RDWR | O_CREAT, 0777);
// check for errors in file opening
if(fd == -1){
printf("Error Creating/Opening File!!\n");
exit(1);
}

int bytesWritten = 0;

// read the messages from the client
while(read(C_sockfd, &fromClient, 1) == 1){
// send the messaged to the copied file
write(fd, &fromClient, 1);
bytesWritten++;
}

printf("The Server Wrote %d bytes!\n", bytesWritten);

// close the connection
close(C_sockfd);
}

}
```

Image For Client


ashhar@Ashhars-Lenovo-B590: ~/Desktop/Systems Assignment 4

```
ashhar@Ashhars-Lenovo-B590:~/Desktop/Systems Assignment 4$ ls -l
total 104
-rwxrwxr-x 1 ashhar ashhar 16432 May 17 22:56 client
-rw-rw-r-- 1 ashhar ashhar  1088 May 18 23:16 q1_client.c
-rw-rw-r-- 1 ashhar ashhar  1962 May 18 23:17 q1_server.c
-rwxrwxr-x 1 ashhar ashhar 16664 May 17 12:07 q2consumer
-rw-rw-r-- 1 ashhar ashhar  3084 May 17 12:06 q2_consumer.c
-rwxrwxr-x 1 ashhar ashhar 16568 May 17 12:07 q2producer
-rw-rw-r-- 1 ashhar ashhar  2423 May 17 12:12 q2_producer.c
-rw-rw-r-- 1 ashhar ashhar  2178 May 17 21:18 q3.c
-rwxrwxr-x 1 ashhar ashhar 16512 May 17 22:48 server
-rw-rw-r-- 1 ashhar ashhar    24 May 17 11:12 test.txt
ashhar@Ashhars-Lenovo-B590:~/Desktop/Systems Assignment 4$ gcc q1_client.c -o client
ashhar@Ashhars-Lenovo-B590:~/Desktop/Systems Assignment 4$ ./client
Error Opening File!
ashhar@Ashhars-Lenovo-B590:~/Desktop/Systems Assignment 4$ ./client test.txt
The Client Wrote 24 bytes
ashhar@Ashhars-Lenovo-B590:~/Desktop/Systems Assignment 4$ ls -l
total 104
-rwxrwxr-x 1 ashhar ashhar 16384 May 18 23:17 client
-rwxrwxr-x 1 ashhar ashhar    24 May 18 23:17 copyFile.txt
-rw-rw-r-- 1 ashhar ashhar  1088 May 18 23:16 q1_client.c
-rw-rw-r-- 1 ashhar ashhar  1962 May 18 23:17 q1_server.c
-rwxrwxr-x 1 ashhar ashhar 16664 May 17 12:07 q2consumer
-rw-rw-r-- 1 ashhar ashhar  3084 May 17 12:06 q2_consumer.c
-rwxrwxr-x 1 ashhar ashhar 16568 May 17 12:07 q2producer
-rw-rw-r-- 1 ashhar ashhar  2423 May 17 12:12 q2_producer.c
-rw-rw-r-- 1 ashhar ashhar  2178 May 17 21:18 q3.c
-rwxrwxr-x 1 ashhar ashhar 16512 May 18 23:17 server
-rw-rw-r-- 1 ashhar ashhar    24 May 17 11:12 test.txt
ashhar@Ashhars-Lenovo-B590:~/Desktop/Systems Assignment 4$ cat copyFile.txt
hello my name is ashhar
ashhar@Ashhars-Lenovo-B590:~/Desktop/Systems Assignment 4$
```

Image For Server


ashhar@Ashhars-Lenovo-B590: ~/Desktop/Systems Assignment 4

```
ashhar@Ashhars-Lenovo-B590:~/Desktop/Systems Assignment 4$ ls -l
total 100
-rwxrwxr-x 1 ashhar ashhar 16384 May 18 23:17 client
-rw-rw-r-- 1 ashhar ashhar  1088 May 18 23:16 q1_client.c
-rw-rw-r-- 1 ashhar ashhar  1962 May 18 23:17 q1_server.c
-rwxrwxr-x 1 ashhar ashhar 16664 May 17 12:07 q2consumer
-rw-rw-r-- 1 ashhar ashhar  3084 May 17 12:06 q2_consumer.c
-rwxrwxr-x 1 ashhar ashhar 16568 May 17 12:07 q2producer
-rw-rw-r-- 1 ashhar ashhar  2423 May 17 12:12 q2_producer.c
-rw-rw-r-- 1 ashhar ashhar  2178 May 17 21:18 q3.c
-rwxrwxr-x 1 ashhar ashhar 16512 May 17 22:48 server
-rw-rw-r-- 1 ashhar ashhar    24 May 17 11:12 test.txt
ashhar@Ashhars-Lenovo-B590:~/Desktop/Systems Assignment 4$ gcc q1_server.c -o server
ashhar@Ashhars-Lenovo-B590:~/Desktop/Systems Assignment 4$ ./server
Server 23653 is waiting
The Server Wrote 24 bytes!
Server 23653 is waiting
```

Question 2

Producer

```c
// macros that define the semaphore names
#define SEMAPHORE_NAME_EMPTIES "/ourSemaphore1.dat"
#define SEMAPHORE_NAME_FULLS "/ourSemaphore2.dat"

// structure to store messages in shared memory
#define TEXT_SZ 2048
struct shared_use_st {
char some_text[TEXT_SZ];
};

// import all the required libaries
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <sys/shm.h>
#include <stdio.h>
#include <fcntl.h>
#include <semaphore.h>

int main(){
int running = 1;
void *shared_memory = (void *)0;
struct shared_use_st *shared_stuff;
char buffer[BUFSIZ];
int shmid;

// get the semaphores
sem_t *semIDE;
sem_t *semIDF;
semIDE = sem_open(SEMAPHORE_NAME_EMPTIES, 0);
semIDF = sem_open(SEMAPHORE_NAME_FULLS, 0);

// check for errors in semaphore creation
if (semIDE == SEM_FAILED) {
perror("Can't open semaphore");
exit(EXIT_FAILURE);
}
if (semIDF == SEM_FAILED) {
perror("Can't open semaphore");
exit(EXIT_FAILURE);
}

// create or get the shared memory segment
shmid = shmget((key_t)1234, sizeof(struct shared_use_st), 0666 | IPC_CREAT);
if (shmid == -1) {
fprintf(stderr, "shmget failed\n");
exit(EXIT_FAILURE);
}

// attach the shared memory segment so it can be used by a program
shared_memory = shmat(shmid, (void *)0, 0);
if (shared_memory == (void *)-1) {
fprintf(stderr, "shmat failed\n");
exit(EXIT_FAILURE);
}
printf("Memory attached at %p\n", shared_memory);

shared_stuff = (struct shared_use_st *) shared_memory;

// loop while the user does not input end
while(running) {
// decremnet the empty semaphore (init value 1)
if(sem_wait(semIDE) !=0)
perror("sem_wait failed in producer");

// send input message to the shared memory segment
printf("Enter some text: ");
fgets(buffer, BUFSIZ, stdin);
strncpy(shared_stuff->some_text, buffer, TEXT_SZ);

// if the user enters "end" break the loop
if (strncmp(buffer, "end", 3) == 0) {
running = 0;
}

// increment the full semaphore (init value 0)
if(sem_post(semIDF)!=0)
perror("sem_post failed in producer");
}

// detach the shared memory segment
if (shmdt(shared_memory) == -1) {
fprintf(stderr, "shmdt failed\n");
exit(EXIT_FAILURE);
}

exit(EXIT_SUCCESS);
}
```

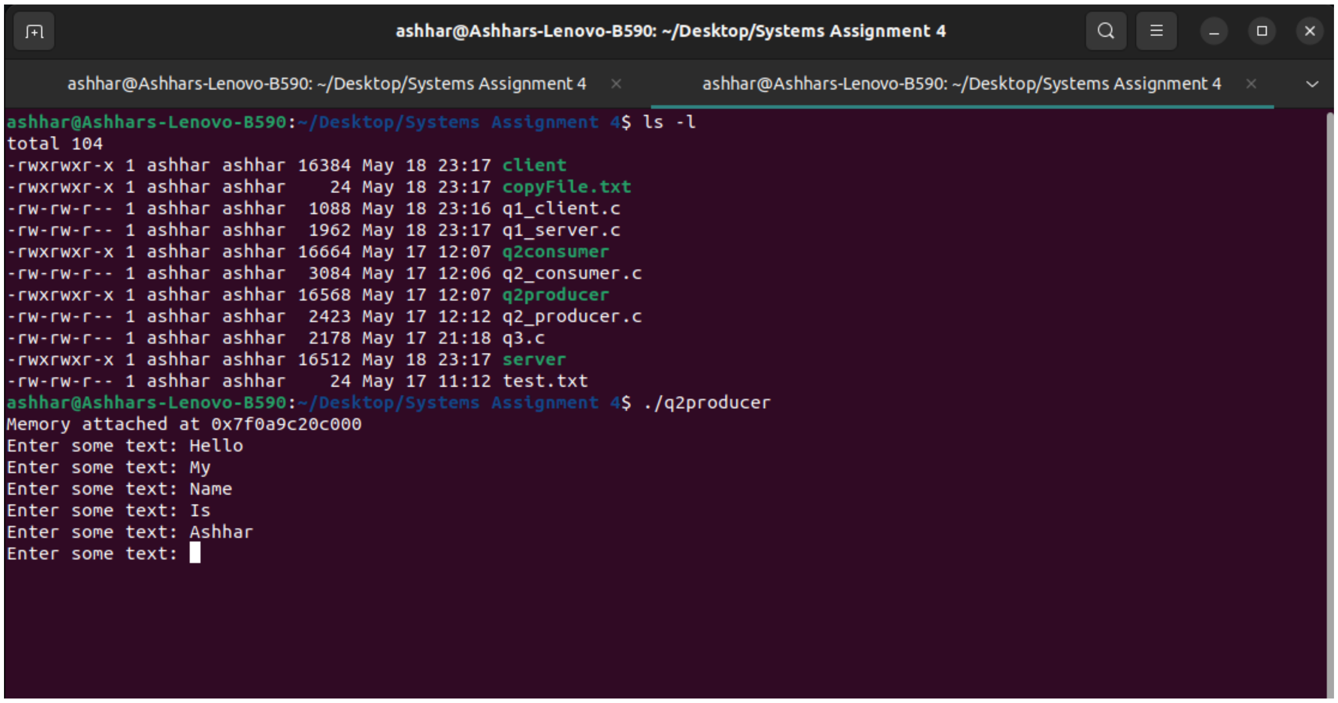
Consumer

```
// structure to store messages in shared memory
#define TEXT_SZ 2048
struct shared_use_st {
char some_text[TEXT_SZ];
};

// macros that define the semaphore names
#define SEMAPHORE_NAME_EMPTIES "/ourSemaphore1.dat"
#define SEMAPHORE_NAME_FULLS "/ourSemaphore2.dat"

// the required libraries
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <sys/shm.h>
#include <stdio.h>
#include <fcntl.h>
#include <semaphore.h>

int main(){ /* We now make the shared memory accessible to the program. */
int running = 1;
void *shared_memory = (void *)0;
struct shared_use_st *shared_stuff;
int shmid;

// initialize the semaphores
sem_t *semIDE;
sem_t *semIDF;
semIDE = sem_open(SEMAPHORE_NAME_EMPTIES, O_CREAT, S_IRUSR|S_IWUSR, 1);
semIDF = sem_open(SEMAPHORE_NAME_FULLS, O_CREAT, S_IRUSR|S_IWUSR, 0);

// check for errors in semaphore initialization
if (semIDE == SEM_FAILED) {
perror("Can't open semaphore");
exit(EXIT_FAILURE);
}
if (semIDF == SEM_FAILED) {
perror("Can't open semaphore");
exit(EXIT_FAILURE);
}

srand((unsigned int)getpid());

// create a shared memory segment
shmid = shmget((key_t)1234, sizeof(struct shared_use_st), 0666 | IPC_CREAT);
if (shmid == -1) {
```

```
fprintf(stderr, "shmget failed\n");
exit(EXIT_FAILURE);
}

// attach the shared memory segment so it is available to the program
shared_memory = shmat(shmid, (void *)0, 0);
if (shared_memory == (void *)-1) {
fprintf(stderr, "shmat failed\n");
exit(EXIT_FAILURE);
}
printf("Memory attached at %p\n", shared_memory);
shared_stuff = (struct shared_use_st *)shared_memory;

// run a loop till we do not encounter an end of file
while(running) {
// decrement the full/producer semaphore (init value 0)
sem_wait(semIDF);
// read the text written by the producer in the shared memory segment
printf("You wrote: %s", shared_stuff->some_text);
// check if we encounter run, if so terminate the loop
if (strncmp(shared_stuff->some_text, "end", 3) == 0) {
running = 0;
}
// increment the empty semaphore (init value 1)
sem_post(semIDE);
}

// close all the shared memory segments and the semaphores
if (shmdt(shared_memory) == -1) {
fprintf(stderr, "shmdt failed\n");
exit(EXIT_FAILURE);
}

if (shmctl(shmid, IPC_RMID, 0) == -1) {
fprintf(stderr, "shmctl(IPC_RMID) failed\n");
exit(EXIT_FAILURE);
}

if(sem_close(semIDE) != 0){
perror("Can't close semaphore");
exit(EXIT_FAILURE);
}

if(sem_close(semIDF) != 0){
perror("Can't close semaphore");
exit(EXIT_FAILURE);
}
```

```
if(sem_unlink(SEMAPHORE_NAME_EMPTIES) != 0){
perror("Can't delete semaphore");
exit(EXIT_FAILURE);
}

if(sem_unlink(SEMAPHORE_NAME_FULLS) != 0){
perror("Can't delete semaphore");
exit(EXIT_FAILURE);
}

exit(EXIT_SUCCESS);
}
```

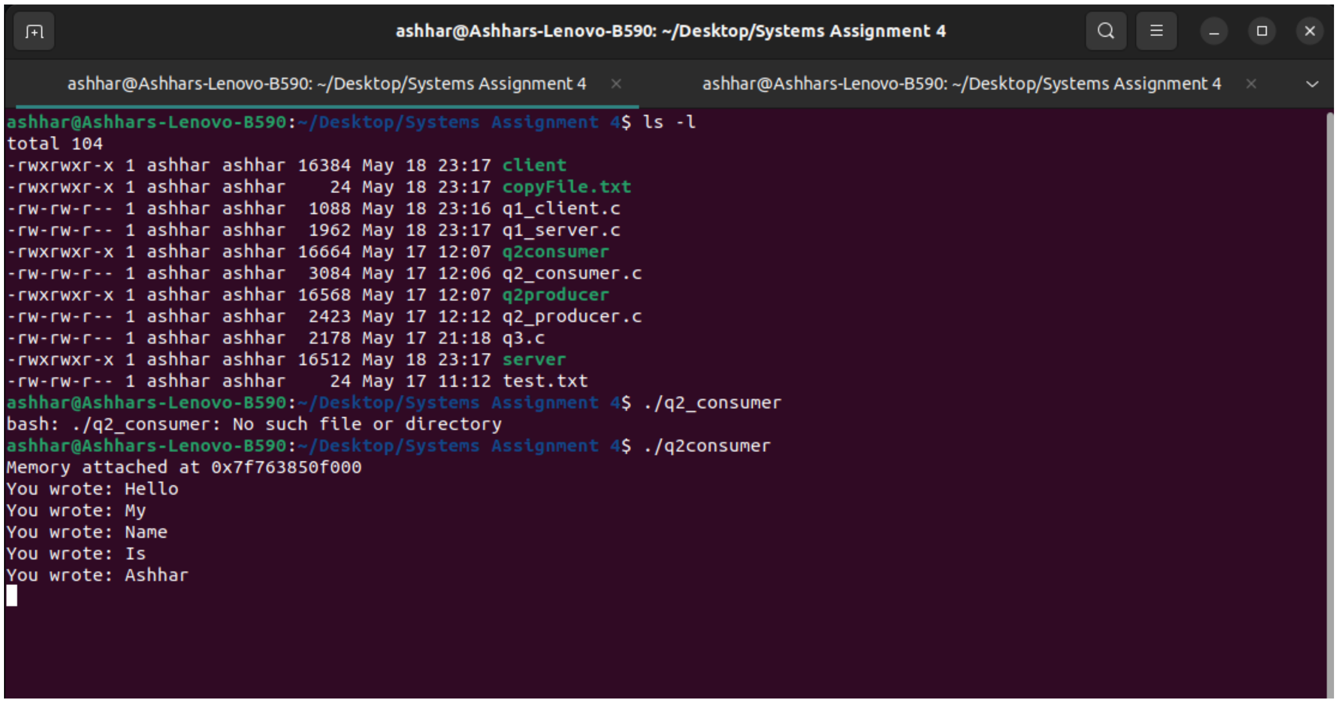
Image For Producer



ashhar@Ashhars-Lenovo-B590: ~/Desktop/Systems Assignment 4

```
ashhar@Ashhars-Lenovo-B590:~/Desktop/Systems Assignment 4$ ls -l
total 104
-rwxrwxr-x 1 ashhar ashhar 16384 May 18 23:17 client
-rwxrwxr-x 1 ashhar ashhar    24 May 18 23:17 copyFile.txt
-rw-rw-r-- 1 ashhar ashhar  1088 May 18 23:16 q1_client.c
-rw-rw-r-- 1 ashhar ashhar  1962 May 18 23:17 q1_server.c
-rwxrwxr-x 1 ashhar ashhar 16664 May 17 12:07 q2consumer
-rw-rw-r-- 1 ashhar ashhar  3084 May 17 12:06 q2_consumer.c
-rwxrwxr-x 1 ashhar ashhar 16568 May 17 12:07 q2producer
-rw-rw-r-- 1 ashhar ashhar  2423 May 17 12:12 q2_producer.c
-rw-rw-r-- 1 ashhar ashhar  2178 May 17 21:18 q3.c
-rwxrwxr-x 1 ashhar ashhar 16512 May 18 23:17 server
-rw-rw-r-- 1 ashhar ashhar    24 May 17 11:12 test.txt
ashhar@Ashhars-Lenovo-B590:~/Desktop/Systems Assignment 4$ ./q2producer
Memory attached at 0x7f0a9c20c000
Enter some text: Hello
Enter some text: My
Enter some text: Name
Enter some text: Is
Enter some text: Ashhar
Enter some text: 
```

Image For Consumer



ashhar@Ashhars-Lenovo-B590: ~/Desktop/Systems Assignment 4

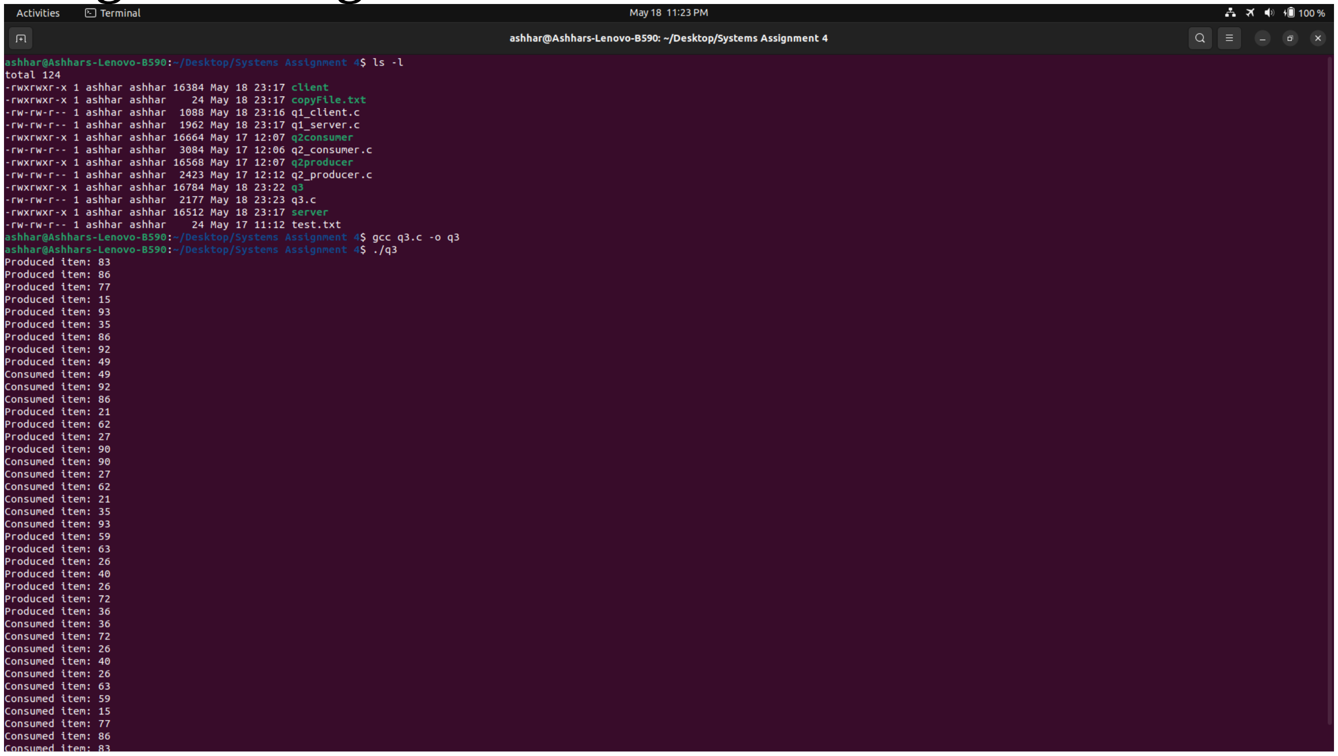
```
ashhar@Ashhars-Lenovo-B590:~/Desktop/Systems Assignment 4$ ls -l
total 104
-rwxrwxr-x 1 ashhar ashhar 16384 May 18 23:17 client
-rwxrwxr-x 1 ashhar ashhar    24 May 18 23:17 copyFile.txt
-rw-rw-r-- 1 ashhar ashhar  1088 May 18 23:16 q1_client.c
-rw-rw-r-- 1 ashhar ashhar  1962 May 18 23:17 q1_server.c
-rwxrwxr-x 1 ashhar ashhar 16664 May 17 12:07 q2consumer
-rw-rw-r-- 1 ashhar ashhar  3084 May 17 12:06 q2_consumer.c
-rwxrwxr-x 1 ashhar ashhar 16568 May 17 12:07 q2producer
-rw-rw-r-- 1 ashhar ashhar  2423 May 17 12:12 q2_producer.c
-rw-rw-r-- 1 ashhar ashhar  2178 May 17 21:18 q3.c
-rwxrwxr-x 1 ashhar ashhar 16512 May 18 23:17 server
-rw-rw-r-- 1 ashhar ashhar    24 May 17 11:12 test.txt
ashhar@Ashhars-Lenovo-B590:~/Desktop/Systems Assignment 4$ ./q2_consumer
bash: ./q2_consumer: No such file or directory
ashhar@Ashhars-Lenovo-B590:~/Desktop/Systems Assignment 4$ ./q2consumer
Memory attached at 0x7f763850f000
You wrote: Hello
You wrote: My
You wrote: Name
You wrote: Is
You wrote: Ashhar
```

Question 3

```
// include all the required libararies
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>

// define the macros to be used
#define BUFFER_SIZE 20
#define NUM_ITEMS 20

// create the shared buffer for both threads
int buffer[BUFFER_SIZE];
int buffer_index = 0;

// create semaphores
sem_t full_sem, empty_sem;
pthread_mutex_t mutex;

// the producer function
void* producer(void* arg) {
// generate random items and put into buffer
int item;
for (int i = 0; i < NUM_ITEMS; ++i) {
item = rand() % 100; // Generate a random item

sem_wait(&empty_sem); // Wait for an empty slot in the buffer
pthread_mutex_lock(&mutex); // Acquire the mutex

// Produce item and add it to the buffer
buffer[buffer_index] = item;
printf("Produced item: %d\n", item);
buffer_index++;

pthread_mutex_unlock(&mutex); // Release the mutex
sem_post(&full_sem); // Signal that the buffer has one more item
}

return NULL;
}

void* consumer(void* arg) {
int item;

for (int i = 0; i < NUM_ITEMS; ++i) {
sem_wait(&full_sem); // Wait for a filled slot in the buffer
pthread_mutex_lock(&mutex); // Acquire the mutex
```

```c
// Consume item from the buffer
buffer_index--;
item = buffer[buffer_index];
printf("Consumed item: %d\n", item);

pthread_mutex_unlock(&mutex); // Release the mutex
sem_post(&empty_sem); // Signal that the buffer has one more empty slot
}

return NULL;
}

int main() {
pthread_t producer_thread, consumer_thread;

// Initialize semaphores and mutex
sem_init(&full_sem, 0, 0);
sem_init(&empty_sem, 0, BUFFER_SIZE);
pthread_mutex_init(&mutex, NULL);

// Create producer and consumer threads
pthread_create(&producer_thread, NULL, producer, NULL);
pthread_create(&consumer_thread, NULL, consumer, NULL);

// Wait for threads to finish
pthread_join(producer_thread, NULL);
pthread_join(consumer_thread, NULL);

// Clean up resources
sem_destroy(&full_sem);
sem_destroy(&empty_sem);
pthread_mutex_destroy(&mutex);

return 0;
}
```

Image For Program



```
ashhar@Ashhars-Lenovo-B590:~/Desktop/Systems Assignment 4$ ls -l
total 124
-rwxrwxr-x 1 ashhar ashhar 16384 May 18 23:17 client
-rwxrwxr-x 1 ashhar ashhar    24 May 18 23:17 copyFile.txt
-rw-rw-r-- 1 ashhar ashhar  1088 May 18 23:16 q1_client.c
-rw-rw-r-- 1 ashhar ashhar  1962 May 18 23:17 q1_server.c
-rwxrwxr-x 1 ashhar ashhar 16664 May 17 12:07 q2consumer
-rw-rw-r-- 1 ashhar ashhar  3084 May 17 12:06 q2_consumer.c
-rwxrwxr-x 1 ashhar ashhar 16568 May 17 12:07 q2producer
-rw-rw-r-- 1 ashhar ashhar  2423 May 17 12:12 q2_producer.c
-rwxrwxr-x 1 ashhar ashhar 16784 May 18 23:22 q3
-rw-rw-r-- 1 ashhar ashhar  2177 May 18 23:23 q3.c
-rwxrwxr-x 1 ashhar ashhar 16512 May 18 23:17 server
-rw-rw-r-- 1 ashhar ashhar    24 May 17 11:12 test.txt
ashhar@Ashhars-Lenovo-B590:~/Desktop/Systems Assignment 4$ gcc q3.c -o q3
ashhar@Ashhars-Lenovo-B590:~/Desktop/Systems Assignment 4$ ./q3
Produced item: 83
Produced item: 86
Produced item: 77
Produced item: 15
Produced item: 93
Produced item: 35
Produced item: 86
Produced item: 92
Produced item: 49
Consumed item: 49
Consumed item: 92
Consumed item: 86
Produced item: 21
Produced item: 62
Produced item: 27
Produced item: 90
Consumed item: 90
Consumed item: 27
Consumed item: 62
Consumed item: 21
Consumed item: 35
Consumed item: 93
Produced item: 59
Produced item: 63
Produced item: 26
Produced item: 40
Produced item: 26
Produced item: 72
Produced item: 36
Consumed item: 36
Consumed item: 72
Consumed item: 26
Consumed item: 40
Consumed item: 26
Consumed item: 63
Consumed item: 59
Consumed item: 15
Consumed item: 77
Consumed item: 86
Consumed item: 83
```