Programming Project

Spring 2022

Course ID: CSE315

Section: 1

Instructor: Mohammad Noor Nabi

Submitted by:

Name: Abu Musa Sakib

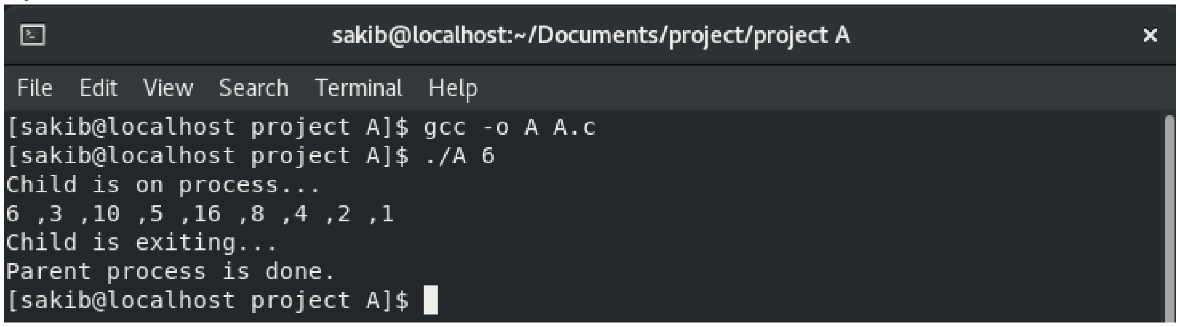
ID: 1810617

Date of Submission: 26 April 2022

Contents

Project A

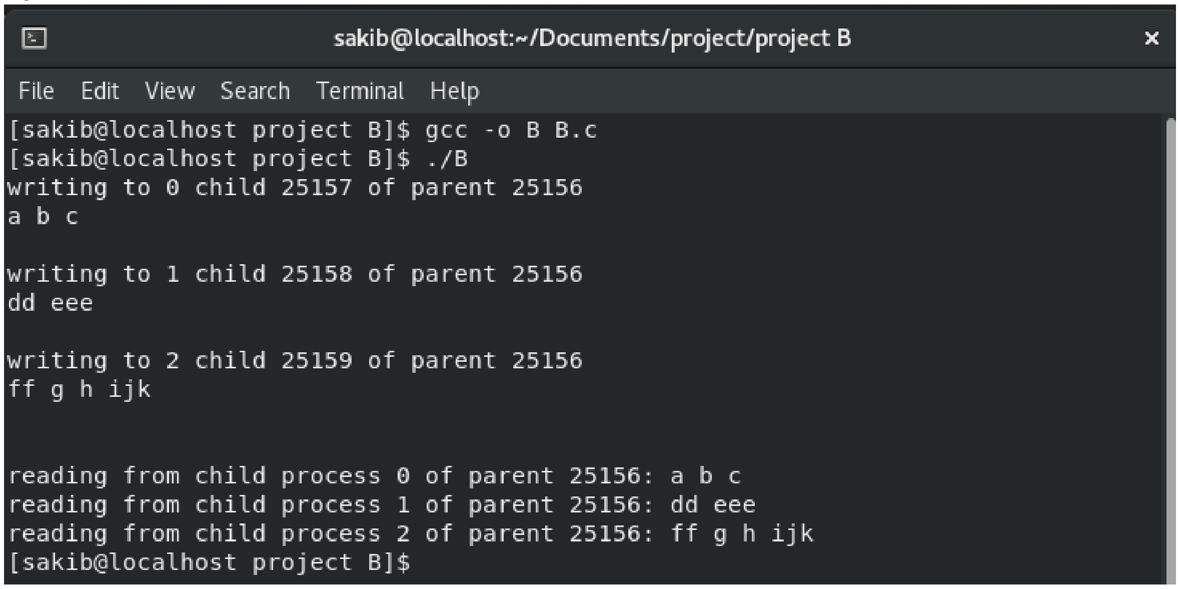
```c
#include <stdio.h>
#include <sys/types.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

void collatz(int n){
   if(fork()==0){
      printf("Child is on process...\n");
      printf("%d ,",n);
      while(n!=1){
         if(n%2==0){
            n/=2;
         }
         else{
            n=3*n+1;
         }
         printf("%d ,",n);
      }
      printf("\b \nChild is exiting...\n");
      exit(0);
   }
   else{
      wait(NULL);
      printf("Parent process is done.\n");
   }
}

int main(int argc, char *argv[]){
      if(argc!=2 || atoi(argv[1])<1){
         printf("Enter valid input (n>0)\n");
      }
      else{
         collatz(atoi(argv[1]));
   }
}
/*
gcc -o A A.c
./A 6
*/
```


sakib@localhost:~/Documents/project/project A
File Edit View Search Terminal Help

```
[sakib@localhost project A]$ gcc -o A A.c
[sakib@localhost project A]$ ./A 6
Child is on process...
6 ,3 ,10 ,5 ,16 ,8 ,4 ,2 ,1
Child is exiting...
Parent process is done.
[sakib@localhost project A]$
```

Project B

```
#include <sys/ipc.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

int main() {
     int n = 3;
     int fd[2 * n];
     char write_msg[n][100];
     char read_msg[100];
     int pid;
     for (int i = 0; i < n; i++) {
          if (pipe(&fd[2 * i]) == -1) {
               printf("Error creating pipe\n");
               return -1;
          }
     }
     for (int i = 0; i < n; i++) {
          if (pid = fork() == 0) {
               printf("writing to %d child %d of parent %d\n", i, getpid(),
getppid());
               close(fd[2 * i]);
               int j = 0;
               int counter = 0;
               while (1) {
                    char in;
                    scanf("%c", &in);
                    if (in == '\n') {
                         counter++;
                         if (counter == 2) {
                              break;
                         }
                    } else {
                         write_msg[i][j] = in;
                         j++;
                    }
               }
               write_msg[i][j] = '\0';
               write(fd[2 * i + 1], &write_msg[i], sizeof(write_msg[i]));
               close(fd[2 * i + 1]);
               exit(0);
          }
          else {
                wait(&pid);
            }
        }
    for (int i = 0; i < n; i++) {
        read(fd[2 * i], read_msg, sizeof(read_msg));
        close(fd[2 * i]);
        printf("\nreading from child process %d of parent %d: %s", i,
getpid(), read_msg);
    }
    printf("\n");
}
/*
gcc -o B B.c
./B
a b c


dd eee


ff g h ijk



*/
```


sakib@localhost:~/Documents/project/project B
File Edit View Search Terminal Help

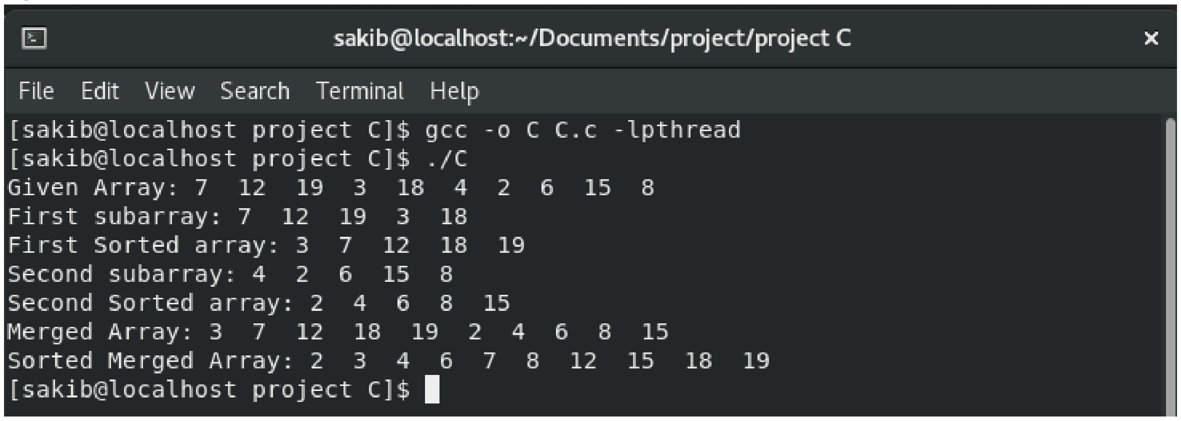
```
[sakib@localhost project B]$ gcc -o B B.c
[sakib@localhost project B]$ ./B
writing to 0 child 25157 of parent 25156
a b c

writing to 1 child 25158 of parent 25156
dd eee

writing to 2 child 25159 of parent 25156
ff g h ijk


reading from child process 0 of parent 25156: a b c
reading from child process 1 of parent 25156: dd eee
reading from child process 2 of parent 25156: ff g h ijk
[sakib@localhost project B]$
```

Project C

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

```c
int arr1[50] = {7, 12, 19, 3, 18, 4, 2, 6, 15, 8}, arr2[50], arr3[50],
arr4[50];
int subarr1, subarr2, total;

void *subarr1_func(void* arg) {
     sleep(1);
     printf("\nFirst subarray: ");
     for (int i = 0; i < subarr1; i++) {
          printf("%d  ", arr2[i]);
     }
     for (int i = 0; i < subarr1; i++) {
          for (int j = 0; j < subarr1 - (i + 1); j++) {
               if (arr2[j] > arr2[j + 1]) {
                    int temp = arr2[j];
                    arr2[j] = arr2[j + 1];
                    arr2[j + 1] = temp;
               }
          }
     }
     printf("\nFirst Sorted array: ");
     for (int i = 0; i < subarr1; i++) {
          printf("%d  ", arr2[i]);
     }
}

void *subarr2_func(void* arg) {
     sleep(2);
     printf("\nSecond subarray: ");
     for (int i = 0; i < subarr2; i++) {
          printf("%d  ", arr3[i]);
     }
     for (int i = 0; i < subarr2; i++) {
          for (int j = 0; j < subarr2 - (i + 1); j++) {
               if (arr3[j] > arr3[j + 1]) {
                    int temp = arr3[j];
                    arr3[j] = arr3[j + 1];
                    arr3[j + 1] = temp;
               }
          }
     }

     printf("\nSecond Sorted array: ");
     for (int i = 0; i < subarr2; i++) {
          printf("%d  ", arr3[i]);
     }
}

void *merge_func(void* arg) {
     sleep(3);
     total = subarr1 + subarr2;
     for (int i = 0; i < subarr1; i++) {
```

```
		arr4[i] = arr2[i];
	}
	int tempsubarr1 = subarr1;
	for (int i = 0; i < subarr2; i++) {
		arr4[tempsubarr1] = arr3[i];
		tempsubarr1++;
	}
	printf("\nMerged Array: ");
	for (int i = 0; i < total; i++) {
		printf("%d  ", arr4[i]);
	}
	for (int i = 0; i < total; i++) {
		for (int j = 0; j < total - i - 1; j++) {
			if (arr4[j + 1] < arr4[j]) {
				int temp = arr4[j];
				arr4[j] = arr4[j + 1];
				arr4[j + 1] = temp;
			}
		}
	}
}

int main(int argc, char const *argv[]) {
	int n = 10;
	pthread_t t1, t2, t3;
	/*
	printf("Enter size of array: ");
	scanf("%d",&n);
	for (int i = 0; i < n; i++){
		scanf("%d",&arr1[i]);
	}*/
	printf("Given Array: ");
	for (int i = 0; i < n; i++) {
		printf("%d  ", arr1[i]);
	}
	int j = 0;
	for (int i = 0; i < n / 2; i++) {
		arr2[j] = arr1[i];
		j++;
	}
	subarr1 = j;
	int k = 0;
	for (int i = n / 2; i < n; i++) {
		arr3[k] = arr1[i];
		k++;
	}
	subarr2 = k;
	pthread_create(&t1, NULL, subarr1_func, NULL);
	pthread_create(&t2, NULL, subarr2_func, NULL);
	pthread_create(&t3, NULL, merge_func, NULL);
	pthread_join(t1, NULL);
	pthread_join(t2, NULL);
	pthread_join(t3, NULL);
	printf("\nSorted Merged Array: ");
	for (int i = 0; i < total; i++) {
		printf("%d  ", arr4[i]);
	}
	printf("\n");
	return 0;

}

/*
gcc -o C C.c -lpthread
./C
10
7 12 19 3 18 4 2 6 15 8
*/
```

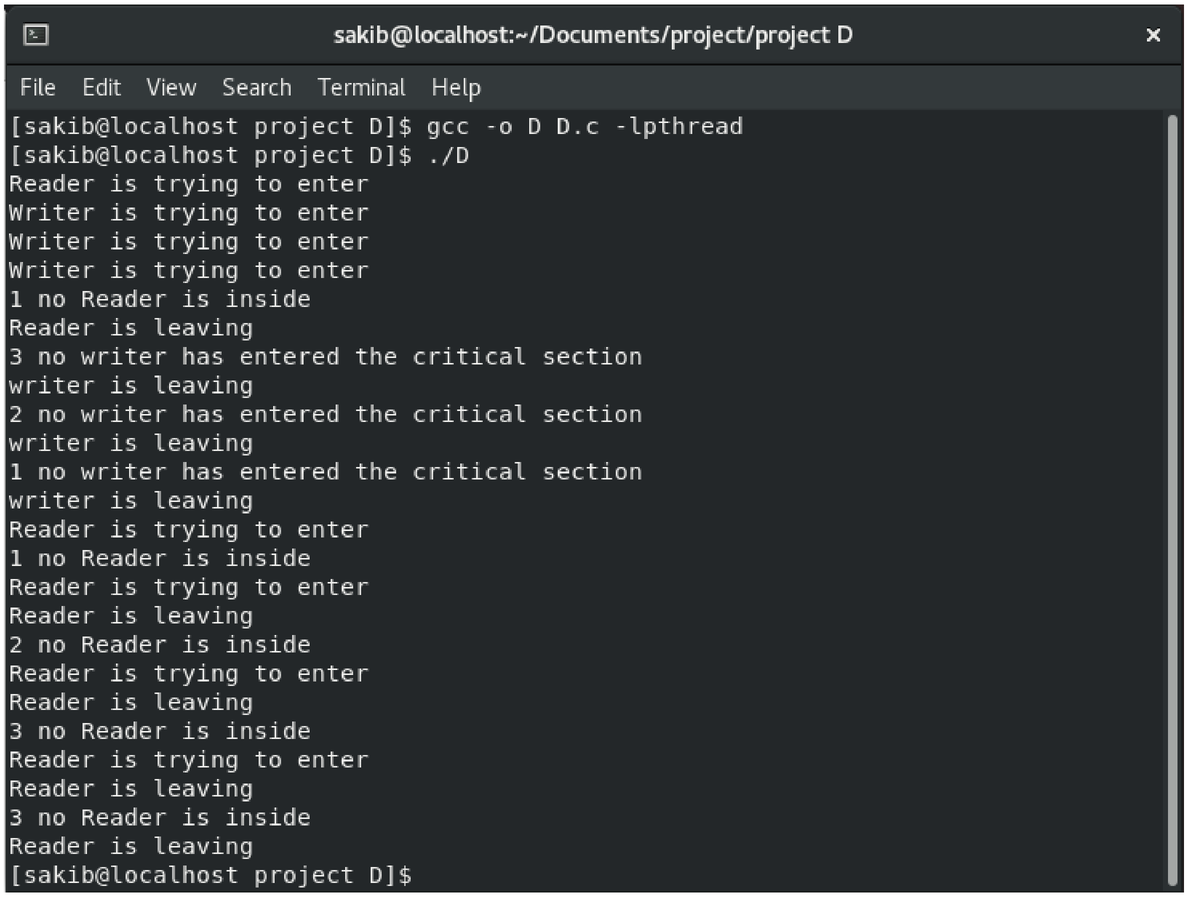

sakib@localhost:~/Documents/project/project C
File Edit View Search Terminal Help

```
[sakib@localhost project C]$ gcc -o C C.c -lpthread
[sakib@localhost project C]$ ./C
Given Array: 7   12   19   3   18   4   2   6   15   8
First subarray: 7   12   19   3   18
First Sorted array: 3   7   12   18   19
Second subarray: 4   2   6   15   8
Second Sorted array: 2   4   6   8   15
Merged Array: 3   7   12   18   19   2   4   6   8   15
Sorted Merged Array: 2   3   4   6   7   8   12   15   18   19
[sakib@localhost project C]$
```

Project D

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

sem_t x, y, z , rsem, wsem;
int readcount, writecount;

void initialize() {
	sem_init(&rsem, 0, 1);
	sem_init(&wsem, 0, 1);
	sem_init(&x, 0, 1);
	sem_init(&y, 0, 1);
	sem_init(&z, 0, 1);
	readcount = 0;
```

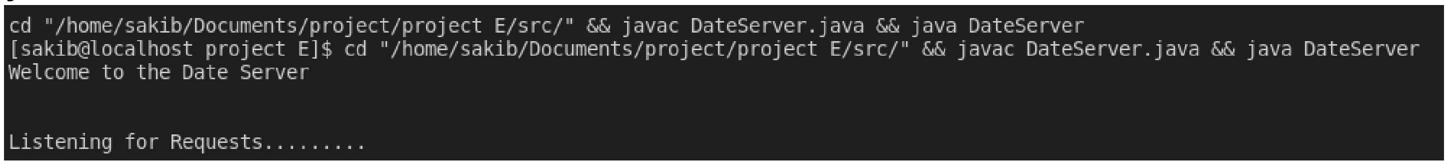
```
    writecount = 0;
}

void* reader(void* arg) {
    sem_wait(&z);
    sem_wait(&rsem);
    sem_wait(&x);
    printf("Reader is trying to enter\n");
    sleep(1);
    readcount++;
    if (readcount == 1) {
        sem_wait(&wsem);
    }
    sem_post(&x);
    sem_post(&rsem);
    sem_post(&z);
    printf("%d no Reader is inside \n", readcount);
    sleep(1);
    printf("Reader is leaving\n");
    sem_wait(&x);
    readcount--;
    if (readcount == 0) {
        sem_post(&wsem);
    }
    sem_post(&x);
}
void* writer(void* arg) {
    printf("Writer is trying to enter\n");
    sleep(1);
    sem_wait(&y);
    writecount++;
    if (writecount == 1) {
        sem_wait(&rsem);
    }
    sem_post(&y);
    sem_wait(&wsem);
    printf("%d no writer has entered the critical section\n", writecount);
    sleep(1);
    printf("writer is leaving\n");
    sem_post(&wsem);
    sem_wait(&y);
    writecount--;
    if (writecount == 0) {
        sem_post(&rsem);
    }
    sem_post(&y);
}

int main()
{
    int r = 5;
    int w = 3;
    pthread_t rtid[r];
    pthread_t wtid[w];
    initialize();
    for (int i = 0; i < r; ++i)
    {
        pthread_create(&(rtid[i]), NULL, &reader, NULL);
    }
    for (int i = 0; i < w; ++i)
    {
        pthread_create(&(wtid[i]), NULL, &writer, NULL);
    }


    for (int i = 0; i < r; ++i)
    {
        pthread_join(rtid[i], NULL);
    }

    for (int i = 0; i < w; ++i)
    {
        pthread_join(wtid[i], NULL);
    }
    return 0;
}

/*
gcc -o D D.c -lpthread
./D
*/
```


sakib@localhost:~/Documents/project/project D
File Edit View Search Terminal Help

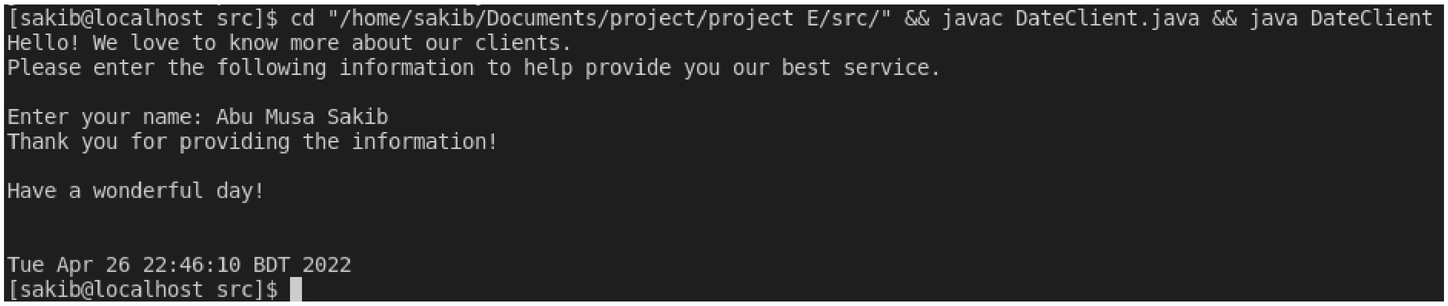
```
[sakib@localhost project D]$ gcc -o D D.c -lpthread
[sakib@localhost project D]$ ./D
Reader is trying to enter
Writer is trying to enter
Writer is trying to enter
Writer is trying to enter
1 no Reader is inside
Reader is leaving
3 no writer has entered the critical section
writer is leaving
2 no writer has entered the critical section
writer is leaving
1 no writer has entered the critical section
writer is leaving
Reader is trying to enter
1 no Reader is inside
Reader is trying to enter
Reader is leaving
2 no Reader is inside
Reader is trying to enter
Reader is leaving
3 no Reader is inside
Reader is trying to enter
Reader is leaving
3 no Reader is inside
Reader is leaving
[sakib@localhost project D]$
```

Project E

```
// Server class
import java.io.*;
import java.net.*;

class Server {
     public static void main(String[] args)
     {
          ServerSocket server = null;

          try {

               // server is listening on port 1234
               server = new ServerSocket(1234);
               //server.setReuseAddress(true);

               // running infinite loop for getting
               // client request
               while (true) {

                    // socket object to receive incoming client
                    // requests
                    Socket c = server.accept();

                    // Displaying that new client is connected
                    // to server
                    System.out.println("New client connected "+
c.getInetAddress().getHostAddress());

                    // create a new thread object
                    ClientHandler clientSock
                         = new ClientHandler(c);

                    // This thread will handle the client
                    // separately
                    new Thread(clientSock).start();
               }
          }
          catch (IOException e) {
               e.printStackTrace();
          }
          finally {
               if (server != null) {
                    try {
                         server.close();
                    }
                    catch (IOException e) {
                         e.printStackTrace();
                    }
               }
          }
     }

// ClientHandler class
     private static class ClientHandler implements Runnable {
          private final Socket clientSocket;

          // Constructor
          public ClientHandler(Socket socket)
          {
               this.clientSocket = socket;
          }

          public void run()
          {
               PrintWriter out = null;
               BufferedReader in = null;
               try {

                    // get the outputstream of client
                    out = new PrintWriter(
                         clientSocket.getOutputStream(), true);
                    // get the inputstream of client
                    in = new BufferedReader(
                         new InputStreamReader(
                              clientSocket.getInputStream()));

                    String line;
                    while ((line = in.readLine()) != null) {

                         // writing the received message from
                         // client
                         System.out.printf(
                              " Sent from the client: %s\n",
                              line);
                                       if("exit".equals(line)){
                                           System.out.println("Client
Disconnected \n");
                                           out.println("you are
disconnected \n");
                                       }else{
                         out.println(line);
                                       }

                    }
               }
               catch (IOException e) {
                    e.printStackTrace();
               }
               finally {
                    try {
                         if (out != null) {
                              out.close();
                         }
                         if (in != null) {
                              in.close();
                              clientSocket.close();
                         }
                    }
                    catch (IOException e) {
                         e.printStackTrace();
                    }
               }
          }
     }
}

// Client class
import java.io.*;
import java.net.*;
import java.util.*;

class Client {
    // driver code
    public static void main(String[] args)
    {
        // establish a connection by providing host and port
        // number
        try (Socket socket = new Socket("localhost", 1234)) {

            // writing to server
            PrintWriter out = new PrintWriter(
                socket.getOutputStream(), true);

            // reading from server
            BufferedReader in
                = new BufferedReader(new InputStreamReader(
                    socket.getInputStream()));

            // object of scanner class
            Scanner sc = new Scanner(System.in);
            String line = null;

            while (!"exit".equalsIgnoreCase(line)) {

                // reading from user
                line = sc.nextLine();

                // sending the user input to server
                out.println(line);
                out.flush();

                // displaying server reply
                System.out.println("Server replied "
                                    + in.readLine());
            }

            // closing the scanner object
            sc.close();
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```



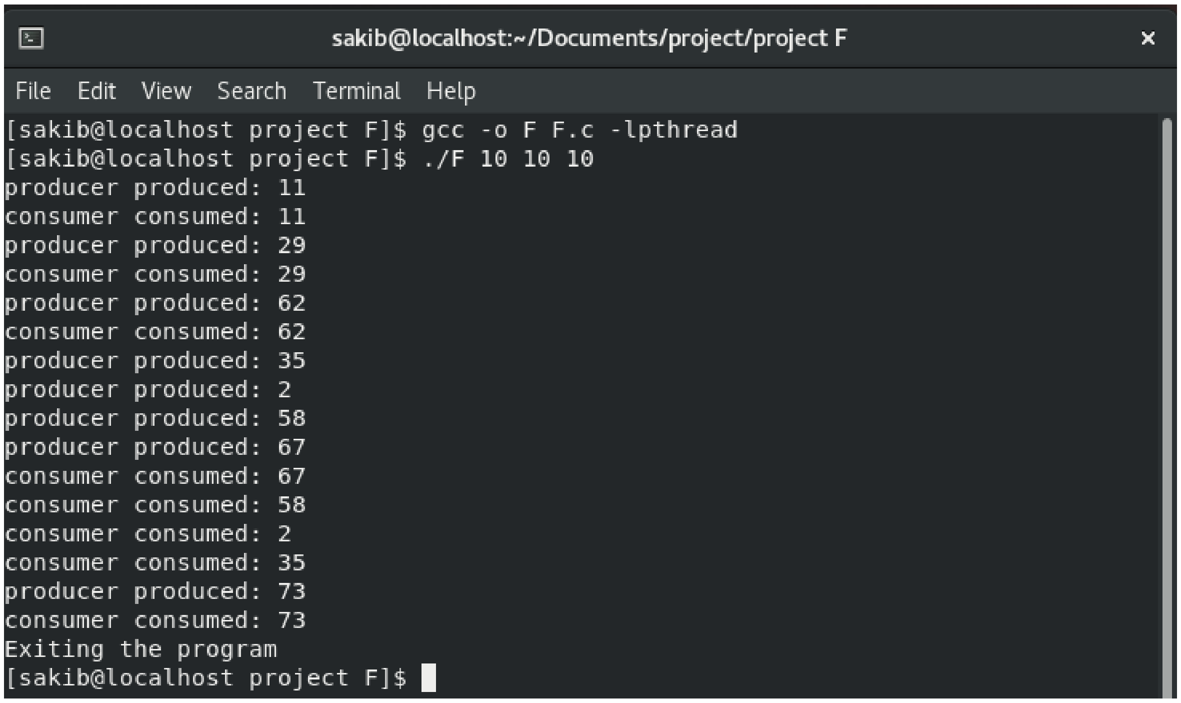
```
cd "/home/sakib/Documents/project/project E/src/" && javac DateServer.java && java DateServer
[sakib@localhost project E]$ cd "/home/sakib/Documents/project/project E/src/" && javac DateServer.java && java DateServer
Welcome to the Date Server


Listening for Requests.........
```


```
[sakib@localhost src]$ cd "/home/sakib/Documents/project/project E/src/" && javac DateClient.java && java DateClient
Hello! We love to know more about our clients.
Please enter the following information to help provide you our best service.

Enter your name: Abu Musa Sakib
Thank you for providing the information!

Have a wonderful day!


Tue Apr 26 22:46:10 BDT 2022
[sakib@localhost src]$
```

Project F

```
//buffer.h
typedef int buffer_item;
#define BUFFER_SIZE 5

//maincode
#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
#include "buffer.h"

pthread_mutex_t mutex;
sem_t full, empty;
buffer_item buffer[BUFFER_SIZE];
int counter;
pthread_t tid;
pthread_attr_t attr;
void *producer(void *param);
void *consumer(void *param);
int insert_item(buffer_item);
int remove_item(buffer_item*) ;


void initializeData() {
     pthread_mutex_init(&mutex, NULL);
     sem_init(&full, 0, 0);
     sem_init(&empty, 0, BUFFER_SIZE);
     pthread_attr_init(&attr);
     counter = 0;
}


void *producer(void *param) {
     buffer_item item;

     while (1) {
          int rNum = rand() / 100000000;
          sleep(rNum);
          item = rand()%100;
          sem_wait(&empty);
          pthread_mutex_lock(&mutex);
          if (insert_item(item)) {
               fprintf(stderr, " Producer report error condition\n");
          }
          else {
               printf("producer produced: %d\n", item);
          }
          pthread_mutex_unlock(&mutex);
          sem_post(&full);
     }
}


void *consumer(void *param) {
     buffer_item item;
     while (1) {
          int rNum = rand() / 1000000000;
          sleep(rNum);
          sem_wait(&full);
          pthread_mutex_lock(&mutex);
          if (remove_item(&item)) {
               fprintf(stderr, "Consumer report error condition\n");
          }
          else {
               printf("consumer consumed: %d\n", item);
          }
          pthread_mutex_unlock(&mutex);
          sem_post(&empty);
     }
}


int insert_item(buffer_item item) {
     if (counter < BUFFER_SIZE) {
          buffer[counter] = item;
          counter++;
          return 0;
     }
     else {
          return -1;
     }
}


int remove_item(buffer_item *item) {
     if (counter > 0) {
          *item = buffer[(counter - 1)];
          counter--;
```

```c
          return 0;
     }
     else {
          return -1;
     }
}

int main(int argc, char *argv[]) {
     int i;
      if(argc != 4) {
      fprintf(stderr, "USAGE:./F <INT> <INT> <INT>\n");
      printf("Exiting the program\n");
      exit(0);
   }
     int sleeptime = atoi(argv[1]);
     int numProd = atoi(argv[2]);
     int numCons = atoi(argv[3]);

     initializeData();

     for (i = 0; i < numProd; i++) {
          pthread_create(&tid, &attr, producer, NULL);
     }
     for (i = 0; i < numCons; i++) {
          pthread_create(&tid, &attr, consumer, NULL);
     }
     sleep(sleeptime);
     printf("Exiting the program\n");
     exit(0);
}

/*
gcc -o F F.c -lpthread
./F 10 10 10
*/
```


sakib@localhost:~/Documents/project/project F
File Edit View Search Terminal Help

```
[sakib@localhost project F]$ gcc -o F F.c -lpthread
[sakib@localhost project F]$ ./F 10 10 10
producer produced: 11
consumer consumed: 11
producer produced: 29
consumer consumed: 29
producer produced: 62
consumer consumed: 62
producer produced: 35
producer produced: 2
producer produced: 58
producer produced: 67
consumer consumed: 67
consumer consumed: 58
consumer consumed: 2
consumer consumed: 35
producer produced: 73
consumer consumed: 73
Exiting the program
[sakib@localhost project F]$ 
```