**Logo

Description automatically generated**

**Programming Project**

**Spring 2022**

**Course ID:** CSE315

**Section:** 1

**Instructor:** Mohammad Noor Nabi

**Submitted by:**

Name: Azwad Fawad Hasan

ID: 2020222

Contents

[Project A 2](#_Toc101905805)

[Project B 3](#_Toc101905806)

[Project C 4](#_Toc101905807)

[Project D 7](#_Toc101905808)

[Project E 10](#_Toc101905809)

[Project F 14](#_Toc101905810)

# Project A

#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

\*/

Text

Description automatically generated

# 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

\*/

Text

Description automatically generated

# Project C

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <pthread.h>

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

\*/

Text

Description automatically generated with medium confidence

# 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

\*/

Text

Description automatically generated

# 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();

}

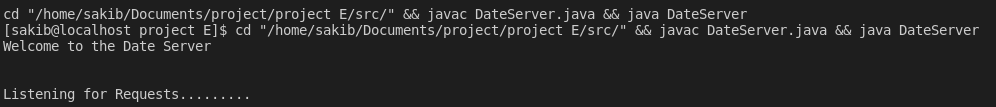
catch (IOException e) {

e.printStackTrace();

}

}

}



Text

Description automatically generated

# 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--;

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

\*/

Text

Description automatically generated