Independent University Bangladesh (IUB)

Course ID: CSE315
Semester: Summer 2021
Section: 1

Submitted To:
Instructor: Mohammad Noor Nabi

Submitted By:
Name: Md. Ashikur Rahman
ID: 1831110

Project

A:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <pthread.h>
struct Params
{
int *start;
size_t len;
int depth;
};
pthread_mutex_t mtx = PTHREAD_MUTEX_INITIALIZER;
// forward declare our thread proc
void *merge_sort_thread(void *pv);
void merge(int *start, int *mid, int *end)
{
int *res = malloc((end - start)*sizeof(*res));
int *lhs = start, *rhs = mid, *dst = res;
while (lhs != mid && rhs != end)
*dst++ = (*lhs <= *rhs) ? *lhs++ : *rhs++;while (lhs != mid)
*dst++ = *lhs++;
while (rhs != end)
*dst++ = *rhs++;
memcpy(start, res, (end - start)*sizeof(*res));
free(res);
}
// our multi-threaded entry point.
```

```
void merge_sort_mt(int *start, size_t len, int depth)

{

if (len < 2)

return;

if (depth <= 0 || len < 4)

{

merge_sort_mt(start, len/2, 0);

merge_sort_mt(start+len/2, len-len/2, 0);

}

else

{

struct Params params = { start, len/2, depth/2 };

pthread_t thrd;

pthread_mutex_lock(&mtx);

printf("Starting subthread...\n");

pthread_mutex_unlock(&mtx);// create our thread

pthread_create(&thrd, NULL, merge_sort_thread, &params);

// recurse into our top-end partition

merge_sort_mt(start+len/2, len-len/2, depth/2);

// join on the launched thread

pthread_join(thrd, NULL);

pthread_mutex_lock(&mtx);

printf("Finished subthread.\n");

pthread_mutex_unlock(&mtx);

}

// merge the partitions.

merge(start, start+len/2, start+len);

}

// our thread-proc that invokes merge_sort. this just passes the

// given parameters off to our merge_sort algorithm
```

```
void *merge_sort_thread(void *pv)
{
struct Params *params = pv;
merge_sort_mt(params->start, params->len, params->depth);
return pv;
}
// public-facing api
void merge_sort(int *start, size_t len)
{merge_sort_mt(start, len, 4); // 4 is a nice number, will use 7 threads.
}
int main()
{
static const unsigned int N = 2048;
int *data = malloc(N * sizeof(*data));
unsigned int i;
srand((unsigned)time(0));
for (i=0; i<N; ++i)
{
data[i] = rand() % 1024;
printf("%4d ", data[i]);
if ((i+1)%8 == 0)
printf("\n");
}
printf("\n");
// invoke our multi-threaded merge-sort
merge_sort(data, N);
for (i=0; i<N; ++i)
{
printf("%4d ", data[i]);
if ((i+1)%8 == 0)
printf("\n");
}
printf("\n");
free(data);return 0;
}
```

Online Code: https://replit.com/@AshikurBitto/CSE315-Project#main.c

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 argc, char const *argv[]){
int n=3;
int fd[2*n];
char write_msg[n][100];
char read_msg[100];
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(fork()==0){
printf("writing child process %d\n",i);
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(NULL);

}

}

for (int i = 0; i < n; i++) {

wait(NULL);

close(fd[2*i+1]);

read(fd[2*i],read_msg,sizeof(read_msg));

close(fd[2*i]);
```

```
printf("\nMessage from child process %d : %s",i,read_msg);
}
printf("\n");
}
/*
gcc -o B B.c
./B
a b c
dd eee
ff g h ijk
*/
```

Online Code: https://replit.com/@AshikurBitto/CSE315-Project-1#main.c

C:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
int arr1[50], arr2[50], arr3[50], arr4[50];
int subarr1, subarr2, total;
void subarr1_func(int *arr1){
sleep(1);
printf("First subarray: ");
for (int i = 0; i < subarr1; i++){
printf("%d ",arr1[i]);
}
for (int i = 0; i < subarr1; i++){
for (int j = 0; j < subarr1-(i+1); j++){
if (arr1[j]>arr1[j+1]){
```

```
int temp=arr1[j];
arr1[j]=arr1[j+1];
arr1[j+1]=temp;
}
}
}
for (int i = 0; i < subarr1; i++){
arr2[i]=arr1[i];
}
printf("\nFirst Sorted array: ");
for (int i = 0; i < subarr1; i++){
printf("%d ",arr2[i]);
}
}
void subarr2_func(int *arr1){
sleep(2);
printf("\nSecond subarray: ");
for (int i = 0; i < subarr1; i++){
printf("%d ",arr1[i]);
}
for (int i = 0; i < subarr1; i++){
for (int j = 0; j < subarr1-(i+1); j++){
if (arr1[j]>arr1[j+1]){
int temp=arr1[j];
arr1[j]=arr1[j+1];
arr1[j+1]=temp;
}
}
}
for (int i = 0; i < subarr1; i++){
```

```
arr3[i]=arr1[i];

}

printf("\nSecond Sorted array: ");

for (int i = 0; i < subarr1; i++){

printf("%d ",arr3[i]);

}

}

void merge_func(int *arr1){

sleep(3);

total=subarr1+subarr2;

for (int i = 0; i < subarr1; i++){

arr1[i]=arr2[i];

}

int tempsubarr1=subarr1;

for (int i = 0; i < subarr2; i++){

arr1[tempsubarr1]=arr3[i];

tempsubarr1++;

}

printf("\nMerged Array: ");

for (int i = 0; i < total; i++){

arr4[i]=arr1[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;

}
}
}
printf("\nSorted Merged Array: ");
for (int i = 0; i < total; i++){
printf("%d ",arr4[i]);
}
printf("\n");
}
int main(int argc, char const *argv[]){
int n;
pthread_t t1,t2,t3;
printf("Enter size of array: ");
scanf("%d",&n);
for (int i = 0; i < n; i++){
scanf("%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,&arr2);
pthread_create(&t2,NULL,subarr2_func,&arr3);
pthread_create(&t3,NULL,merge_func,&arr4);
pthread_join(t1,NULL);
pthread_join(t2,NULL);
pthread_join(t3,NULL);
return 0;
exit(0);
}
/*
gcc -o C C.c -lpthread
./C
10
7 12 19 3 18 4 2 6 15 8
*/
```

Online Code: https://replit.com/@AshikurBitto/CSE315-Project-2#main.c

D:

```
#include<stdio.h>
#include<string.h>
#include<pthread.h>
#include<stdlib.h>
#include<unistd.h>
pthread_mutex_t rd,wrt;
int readcount;
void initialize(){
pthread_mutex_init(&rd,NULL);
pthread_mutex_init(&wrt,NULL);
readcount=0;
}
void* reader(void* arg){
int waittime;
```

```
waittime = rand() % 5;

printf("Reader is trying to enter\n");

pthread_mutex_lock(&rd);

readcount++;

if(readcount==1){

pthread_mutex_lock(&wrt);

}

pthread_mutex_unlock(&rd);

printf("%d Reader is inside \n",readcount);

sleep(waittime);

pthread_mutex_lock(&rd);

readcount--;

if(readcount==0){

pthread_mutex_unlock(&wrt);

}

pthread_mutex_unlock(&rd);

printf("Reader is leaving\n");

}

void* writer(void* arg){

int waittime;

waittime = rand() % 3;

printf("Writer is trying to enter\n");

pthread_mutex_lock(&wrt);

printf("writer has entered the critical section\n");

sleep(waittime);

pthread_mutex_unlock(&wrt);

printf("writer is leaving\n");

}

int main()

{
```

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

Online Code: https://replit.com/@AshikurBitto/CSE315-Project-4#main.c

E:

```
DataServer.java
import java.net.*;
import java.io.*;
public class DataServer
{
public static void main(String[]args)
{
try
}
{
ServerSocket sock = new ServerSocket(6013);
/*now listen for connections*/
while(true){
/**
*Create a client class that extends the Runnable or Thread class
*to add to the Thread object from the DateServer on each connection from
Client*
**/
//For example, client is class which extends the Runnable interface
//and override the run method.
// it is a dummy class
Client c;
socket client = sock.accept();
//then create a client with the socket that is accepted by the
//server
c=new Client(client);
//then create a thread object of thread class using constructor with client as argument
Thread t = new Thread(c);
//call start method
t.start();
printWriter pout = new
printWriter(client.getOutputStream(),true);
/*write the date to the socket */
pout.println(new java.util.Date().toString());
/*close the socket and resumes*/
/* listening for connections */
client.close();
//call stop method to stop execution of thread
t.stop();
}
}
catch(IOException ioe){
System.err.println(ioe);
}
}
}
//Sample client java class that implements the Runnable interface
//client.java
import java.io.BufferdReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.printWriter;
import java.net.Socket;
class Client implements Runnable{
private Socket client;
```

```
//Constructor
Client(Socket client){
this.client = client;
}
//override run method
public void run(){
String line;
BufferedReader in = null;
printWriter out = null;
try{
in = new BufferedReader(new InputStreamReader(client.getInputStream()));
out = new
printWriter(client.getOutputStream(),true);
}
catch(IOException e){
System.out.println("failed read or write");
System.exit(-1);
}
while(true){
try{
line = in.readLine();
//Send data back to client
out.println(line);
//print data from the server to console.
System.out.println(out.toString());
}catch(IOException e){
System.out.println("Read failed");
System.exit(-1);
}
}
}

}
```

Online Code: https://replit.com/@AshikurBitto/CSE315-Project-5#Main.java

F

```
#include<iostream>

#include "ProducerConsumer.h"

//static data variable

static int Buffer[BUFFER_SIZE];//the buffer

static int =0;

static int out = 0;

static Semaphore NotFull("NotFull",BUFFER_SIZE);

static Semaphore NotEmpty("NotEmpty",0);

static Semaphore BufferLock("BufferLock",1);

strstream *Filler(int n)

{

int i;

strstream *Space;

Space = new strstream;

for(i=0;i<n;i++)

(*Space)<<'';

(*Space)<<'\0';

return Space;

}

ProducerThread::ProducerThread(int No,int
numberofdata):Number(No),NumberOfData(numberofdata)

{

ThreadName.seekp(0,ios::beg);

ThreadName<<"Producer"<<No<<'\0';
};
/Consumer Thread::ConsumerThread(int No)
:Number(No)
{
ThreadName.seekp(0,ios::beg);
ThreadName<<"Consumer"<<No<<'\0';
}
Void ProducerThread::ThreadFunc()
{
Thread::ThreadFunc();
int data;
strstream *Space;
Space = Filler(4);
for(int i=1;i<=NumberOfData;i++)
Delay();
NotFull.wait();
BufferLock.wait();
data = rand()%100+1000 = Number;
Buffer[In] = data;
cout<<Space->str()<<ThreadName.str()<<"deposited"<<data<<" to the buffer"<<endl;
In = (In+1)%BUFFER_SIZE;
BufferLock.Signal();
NotEmpty.Signal();
}
Delay();
NotFull.Wait();
BufferLock.wait();
Buffer[In] = END;
cout<<Space->str()<<ThreadName.str()<<"deposited END and Exit"<<endl;
In = (In +1)%BUFFER_SIZE;
```

```
BufferLock.Signal();
NotEmpty.Signal();
Exit();
}
void ConsumerThread::ThreadFunc()
{
Thread::ThreadFunc();
int data=0;
strstream *Space;
Space = Filler(2);
while(true){
Delay();
NotEmpty.Wait();
BufferLock.wait();
data = Buffer[out];
if(data !=END){
cout<<Space->str()<<ThreadName.str()<<"received"<<data<<"from the buffer"<<endl;
Out = (Out+1)%BUFFER_SIZE;
BufferLockSignal();
NotFullSignal();
}
else{
cout<<Space->str()<<ThreadName.str()<<"received END and exites"<<endl;
Out = (Out+1)%BUFFER_SIZE;
BufferLock.Signal();
NotFullSignal();
break;
}
}
Exit();
```

```
}
```

Online Code: https://replit.com/@AshikurBitto/CSE315-Project-7#main.cpp