Programming project

Name: Ananya Chakraborty Gourab

ID: 1821503

Course ID: CSE315

Course name: Design of operating system

Sec: 02

Answer to the question no A

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

printf("\n\n Welcome to Collatz conjecture concerns\n");

int n=0;

int k=0;

pid\_t pid;

do

{

printf("Please enter a number: ");

scanf("%d", &k);

}while (k <= 0);

pid = fork();

if (pid == 0)

{

printf("Child is working...\n");

printf("%d\n",k);

while (k!=1)

{

if (k%2 == 0)

{

k = k/2;

}

else if (k%2 == 1)

{

k = 3 \* (k) + 1;

}

printf("%d\n",k);

}

printf("Child process is done.\n");

}

else

{

printf("Parents is waiting on child process...\n");

printf("Parent process is done.\n");

}

return 0;

}

Answer to the question no-B

#include<stdio.h>

#include<unistd.h>

#include<cstdlib>

#include<cstring>

using namespace std;

int\* n;

int main(){

int pipeA[3], a;

int pipeB[3], b;

int pipeC[3], c;

a = pipe(pipeA);

b = pipe(pipeB);

c = pipe(pipeC);

if(a == -1) printf("pipe A failed!\n");

else printf("pipe A made!\n");

if(b == -1) printf("pipe B failed!\n");

else printf("pipe B made!\n");

if(c == -1) printf("pipe C failed!\n");

else printf("pipe C made!\n");

const char \*line0 = "hello\n this is line 0!\n";

const char \*line1 = "hello again. this is line 1\n it's line 1.\n";

const char \*line2 = "hello again. this is line 2\n it's line 2.\n";

int pA = fork();

if(pA==0){ //it's child A

close(pipeA[0]);

for(int i=0; i<strlen(line0); i++){

if(line0[i]=='\n') break;

write(pipeA[1], line0+i, 1);

}

close(pipeA[1]);

exit(0);

}

else{ // its the parent

int pB = fork();

if(pB==0){ // it's child B

close(pipeB[0]);

for(int i=0; i<strlen(line1); i++){

if(line1[i]=='\n') break;

write(pipeB[1], line1+i, 1);

}

close(pipeB[1]);

exit(0);

}

else{ // its the parent

int pC = fork();

if(pC==0){ // it's child c

close(pipeC[0]);

for(int i=0; i<strlen(line2); i++){

if(line1[i]=='\n') break;

write(pipeC[1], line2+i, 1);

}

close(pipeC[1]);

exit(0);

}

else{ // its the parent

close(pipeA[1]);

close(pipeB[1]);

close(pipeC[1]);

char buf;

printf("[parent] reading pipe A\n");

while(read(pipeA[0], &buf, 1) > 0) printf("%c",buf);

close(pipeA[0]);

printf("\n");

printf("[parent] reading pipe B\n");

while(read(pipeB[0], &buf, 1) > 0) printf("%c",buf);

close(pipeB[0]);

printf("\n");

printf("[parent] reading pipe C\n");

while(read(pipeC[0], &buf, 1) > 0) printf("%c",buf);

close(pipeC[0]);

}

}

exit(0);

}

}

Answer to the question no-C

#include<semaphore.h>

#include<pthread.h>

#include<stdio.h>

int rc=0,wc=0,val;

pthread\_mutex\_t mutex1,mwrite,mread,rallow;

pthread\_t tr1,tr2,tw1,tw2;

pthread\_attr\_t tr1attr,tr2attr,tw1attr,tw2attr;

void \*writer();

void \*reader();

int main()

{

pthread\_mutex\_init(&mwrite,NULL);

pthread\_mutex\_init(&mread,NULL);

pthread\_mutex\_init(&rallow,NULL);

pthread\_mutex\_init(&mutex1,NULL);

pthread\_attr\_init(&tw1attr);

pthread\_attr\_init(&tr1attr);

pthread\_attr\_init(&tr2attr);

pthread\_attr\_init(&tw2attr);

printf("\n Writer 1 created");

pthread\_create(&tw1,&tw1attr,writer,NULL);

printf("\n Reader 1 created");

pthread\_create(&tr1,&tr1attr,reader,NULL);

printf("\n Reader 2 created");

pthread\_create(&tr2,&tr2attr,reader,NULL);

printf("\n WRITER 2 created");

pthread\_create(&tw2,&tw2attr,writer,NULL);

pthread\_join(tw1,NULL);

pthread\_join(tr1,NULL);

pthread\_join(tr2,NULL);

pthread\_join(tw2,NULL);

return 0;

}

void \*writer()

{

pthread\_mutex\_lock(&mwrite);

wc++;

if(wc==1)

pthread\_mutex\_lock(&rallow);

pthread\_mutex\_unlock(&mwrite);

pthread\_mutex\_lock(&mutex1);

printf("\n Enter data in writer %d",wc);

scanf("%d",&val);

pthread\_mutex\_unlock(&mutex1);

pthread\_mutex\_lock(&mwrite);

wc--;

if(wc==0)

pthread\_mutex\_unlock(&rallow);

pthread\_mutex\_unlock(&mwrite);

pthread\_exit(0);

}

void \*reader()

{

pthread\_mutex\_lock(&rallow);

pthread\_mutex\_lock(&mread);

rc++;

if(rc==1)

pthread\_mutex\_lock(&mutex1);

pthread\_mutex\_unlock(&mread);

pthread\_mutex\_unlock(&rallow);

printf("\n reader %d read data: %d",rc,val);

pthread\_mutex\_lock(&mread);

rc--;

if(rc==0)

pthread\_mutex\_unlock(&mutex1);

pthread\_mutex\_unlock(&mread);

pthread\_exit(0);

}

Answer to the question no-D

#include<stdio.h>

#include<string.h>

#include<pthread.h>

#include<stdlib.h>

#include<unistd.h>

pthread\_mutex\_t rd,wrt;//reader and writer mutex

int readcount;//storing the number of reader

void initialize(){

pthread\_mutex\_init(&rd,NULL);

pthread\_mutex\_init(&wrt,NULL);

readcount=0;

}

void\* reader(void\* arg){

int waittime;

waittime = rand() % 5;

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

pthread\_mutex\_lock(&rd);

readcount++;

if(readcount==1){

pthread\_mutex\_lock(&wrt);

}

pthread\_mutex\_unlock(&rd);

printf("\n%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("\nReader is leaving\n");

}

void\* writer(void\* arg){

int waittime;

waittime = rand() % 3;

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

pthread\_mutex\_lock(&wrt);

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

sleep(waittime);

pthread\_mutex\_unlock(&wrt);

printf("\nwriter is leaving\n");

}

int main()

{

int r = 5;//number of reader

int w = 3;//number of writer

pthread\_t rtid[r];

pthread\_t wtid[w];

initialize();

//spawning reader thread

for (int i = 0; i < r; ++i)

{

int err;

err = pthread\_create(&(rtid[i]),NULL,&reader,NULL);

}

//spawning writer thread

for (int i = 0; i < w; ++i)

{

int err;

err = pthread\_create(&(wtid[i]),NULL,&writer,NULL);

}

//exiting thread

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;

}

Answer to the question no-E

import java.net.\*;

import java .io.\*;

public class client{

public static void main(string [] args){

new client();

}

Private final InetAddress host;

Private final socket port;

Private final printstream out;

Private final BufferedReader in;

Private client(){

/\*

\*Get IP address and set port

\*/

InetAddress host = null;

Socket port = null;

try{

host = InetAddress.getByName(“localhost”);

port= new socket(host, 1337);

}catch(IOException e){

e.printstackTace();

}

this.host = host;

this. Port= port;

/\*

\*Get input and output streams

\*/

printStream out = null;

BufferedReader in = null;

try{

out = new printStream(port.getOutputStream());

in= new BufferedReader(new InputstreamReader(port.getInputStream()));

}catch(IOException e){  
e. printStackTrace();

}

this.out=out;

this.in=in;

int read = 0;

do{

try{

read = in.read();

}catch(IOException e)[

e.printStackTrace();

}

System.out.print(char)read);

}while(read!=-1);

}

}

Connection.java

Import java.io.\*;

Import java.net.\*;

Public class connection {

Private server server;

Private socket port;

Private printStream out;

Private BufferedReader in;

Protected Connection (Server server, socket port){

System.out.printIn(“Connection established at “+ port.getInetAddress().getcanonicalHostNmae());

this. Server = server;

this.port = port;

Outputstream out= null;

InputStream in = null;

try{

Out= this.port.getOutputStream();

In=this.port.getInputStream();

}catch(IOException e){

e.printStackTrace();

}

this.out= new printstream(out);

this.in= new BufferedReader(new InputStreamReader(in));

this.out.printIn(“Tittysprinkles”);

}

}

Server.java

import java.net.\*;

import.java.util.ArrayList;

import.java.io.\*;

public class server{

public static void main(string[] args){

new server();

private final InetAddress localhost;

private final serversocket port;

private ArrayList<Connection> clients;

private server server;

private server(){

client = new ArrayList<Connection>();

InetAddress localhost = null;

Serversocket port = null;

Try{

Localhost = InetAddress.getLocalHost();

Port = new serversocket(1337);

}catch(IOException e){

e.printStackTrace();

}

this.localhost = localhost;

this.port= port;

while(this.port.isClosed()){

try{

final socket s = this.port.accept();

new thread{

new runnable(){

@override

Public void run(){

client.add(new Connection(server.s));

}

}

}start();

}catch(IOException e){  
e.printStackTrace();

}

}

}

}

Answer to the question no –F

#include <sys/ipc.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <unistd.h>

int main(){

int processPipe[6];

char writemessage[3][100];

char readmessage[100];

for(int i=0;i<3;i++){

pipe(&processPipe[2\*i]);

}

for (int i = 0; i < 3; i++) {

if(fork()==0){

printf("writing child process %d\n",i);

int j=0;

int counter=0;

while(1){

char in;

scanf("%c",&in);

if(in == '\n'){

counter++;

if(counter==2){

break;

}

}else{

writemessage[i][j]=in;

j++;

}

}

write(processPipe[2\*i+1],writemessage[i],sizeof(writemessage[i]));

exit(0);

}

}

for (int i = 0; i < 3; i++) {

wait(NULL);

read(processPipe[2\*i],readmessage,sizeof(readmessage));

printf("\nMessage from child process %d Message: %s",i,readmessage);

}

}