

OPREATING SYSTEM

NAME: - SAURABH SINGH

SECTION:- K17RG

ROLL NO:- A-14

Question No.14

A university computer science department has a teaching assistant (TA) who helps undergraduate students with their programming assignments during regular office hours. The TA’s office is rather small and has room for only one

Desk with a chair and computer. There are three chairs in the hallway outside the office where students can sit and wait if the TA is currently helping another student. When there are no students who need help during office hours, the

TA sits at the desk and takes a nap. If a student arrives during office hours and finds the TA sleeping, the student must awaken the TA to ask for help. If a student arrives and finds the TA currently helping another student, the student

Sits on one of the chairs in the hallway and waits. If no chairs are available, the student will come back at a later time.

Using POSIX threads, mutex locks, and semaphores, implement a solution that coordinates the activities of the TA and the students.

GitHub link:- https://github.com/saurabhs1998/operating-system-project.git

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <pthread.h>

#include <semaphore.h>

void\* student\_actions( void\* student\_id );

void\* ta\_actions();

#define NUM\_WAITING\_CHAIRS 3

#define DEFAULT\_NUM\_STUDENTS 5

sem\_t sem\_students;

sem\_t sem\_ta;

pthread\_mutex\_t mutex\_thread;

int waiting\_room\_chairs[3];

int number\_students\_waiting = 0;

int next\_seating\_position = 0;

int next\_teaching\_position = 0;

int ta\_sleeping\_flag = 0;

int main( int argc, char \*\*argv ){

int i;

int student\_num;

if (argc > 1 )

{

if ( isNumber( argv[1] ) == 1) {

student\_num = atoi( argv[1] );

}

else {

printf("Invalid input. Quitting program.");

return 0;

}

}

else

{

student\_num = DEFAULT\_NUM\_STUDENTS;

}

int student\_ids[student\_num];

pthread\_t students[student\_num];

pthread\_t ta;

sem\_init( &sem\_students, 0, 0 );

sem\_init( &sem\_ta, 0, 1 );

//Create threads.

pthread\_mutex\_init( &mutex\_thread, NULL );

pthread\_create( &ta, NULL, ta\_actions, NULL );

for( i = 0; i < student\_num; i++ )

{

student\_ids[i] = i + 1;

pthread\_create( &students[i], NULL, student\_actions, (void\*) &student\_ids[i] );

}

//Join threads

pthread\_join(ta, NULL);

for( i =0; i < student\_num; i++ )

{

pthread\_join( students[i],NULL );

}

return 0;

}

void\* ta\_actions() {

printf( "Checking for students.\n" );

while( 1 ) {

//if students are waiting

if ( number\_students\_waiting > 0 ) {

ta\_sleeping\_flag = 0;

sem\_wait( &sem\_students );

pthread\_mutex\_lock( &mutex\_thread );

int help\_time = rand() % 5;

//TA helping student.

printf( "Helping a student for %d seconds. Students waiting = %d.\n", help\_time, (number\_students\_waiting - 1) );

printf( "Student %d receiving help.\n",waiting\_room\_chairs[next\_teaching\_position] );

waiting\_room\_chairs[next\_teaching\_position]=0;

number\_students\_waiting--;

next\_teaching\_position = ( next\_teaching\_position + 1 ) % NUM\_WAITING\_CHAIRS;

sleep( help\_time );

pthread\_mutex\_unlock( &mutex\_thread );

sem\_post( &sem\_ta );

}

//if no students are waiting

else {

if ( ta\_sleeping\_flag == 0 ) {

printf( "No students waiting. Sleeping.\n" );

ta\_sleeping\_flag = 1;

}

}

}

}

void\* student\_actions( void\* student\_id ) {

int id\_student = \*(int\*)student\_id;

int k=4;

while( k-- ) {

//if student is waiting, continue waiting

if ( isWaiting( id\_student ) == 1 ) { continue; }

//student is programming.

int time = rand() % 5;

printf( "\tStudent %d is programming for %d seconds.\n", id\_student, time );

sleep( time );

pthread\_mutex\_lock( &mutex\_thread );

if( number\_students\_waiting < NUM\_WAITING\_CHAIRS ) {

waiting\_room\_chairs[next\_seating\_position] = id\_student;

number\_students\_waiting++;

//student takes a seat in the hallway.

printf( "\t\tStudent %d takes a seat. Students waiting = %d.\n", id\_student, number\_students\_waiting );

next\_seating\_position = ( next\_seating\_position + 1 ) % NUM\_WAITING\_CHAIRS;

pthread\_mutex\_unlock( &mutex\_thread );

//wake TA if sleeping

sem\_post( &sem\_students );

sem\_wait( &sem\_ta );

}

else {

pthread\_mutex\_unlock( &mutex\_thread );

//No chairs available. Student will try later.

printf( "\t\t\tStudent %d will try later.\n",id\_student );

break;

}

break; }

}

int isNumber(char number[])

{

int i;

for ( i = 0 ; number[i] != 0; i++ )

{

if (!isdigit(number[i]))

return 0;

}

return 1;

}

int isWaiting( int student\_id ) {

int i;

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

if ( waiting\_room\_chairs[i] == student\_id ) { return 1; }

}

return 0;

}