Umang Srivastava
2019101090

OSN Assignment 4

Q2 : Back to College

Overview

In this code, company_threads produce a random batch of vaccines when required. The zones_threads check for the availability of vaccines allotted to zones. The student_thread checks for the availability of slots to vaccinate students.

Variables and Data Structures

1. **Composite Variables:**

- **Company**

```
typedef struct Company
{
    int id; // id of the company
    int status; // status of the company - whether it is producing vaccines
or not
    int batches; // number of batches of vaccines produced by the company
    int p; // number of vaccines in each batch produced by the company
    int prob; // probability of success of the vaccine
    pthread_mutex_t company_mutex; // mutex to make sure that only one zone
accesses at a given time
    pthread_t company_thread_id; // thread to produce vaccine
    pthread_cond_t condt_zone;
} Company;
```

- **Zone**

```
typedef struct Zone
{
    int id; // id of the zone
    int p; // capacity of the zone
    int slots; // number of slots in the zone
    int strength; // number of students served by the zone
    int company_supply; // id of the company which supplies the zone
    pthread_mutex_t zone_mutex; // mutex to make sure that only one student
access a zone at a given time
    pthread_t zone_thread_id; // thread to check and fill vaccines in zone
    pthread_cond_t condt_student;
} Zone;
```

- **Student**

```
typedef struct Student
{
    int id; // id of the student
    int round; // number of round of the student
    pthread_t student_thread_id; // thread to wait for slot and allot it to
student once it is available
} Student;
```

2. Arrays

```
Company companies[MAXL]; // array of companies
Zone zones[MAXL]; // array of Vaccination Zones
Student students[MAXL]; // array of Students
```

Functions

1. vaccine_ready

```
void vaccine_ready(Company *c)
{
    while(1>0)
    {
        if(c->batches != 0)
        {
            pthread_cond_wait(&(c->condt_zone), &(c->company_mutex));
        }
        else
        {
            break;
        }
    }

    printf("%sAll the vaccines prepared by Pharmaceutical Company %d are emptied.
Resuming production now.\n\n",MAG,c->id);
    pthread_mutex_unlock(&(c->company_mutex));
}
```

This function basically indicates when the batch produced by the company will be exhausted. I used conditional wait to avoid busy waiting.

2. ready_vaccination_zone

```
void ready_vaccination_zone(Zone *z)
{
    while(1>0)
    {
        if(z->slots == z->strength)
            break;
        else
            pthread_cond_wait(&(z->condt_student), &(z->zone_mutex));
    }
    pthread_mutex_unlock(&(z->zone_mutex));
}
```

If all the slots are filled, it breaks the loop else waits conditionally.

3. student_vaccinated

```c
int student_vaccinated(int i, int id)
{
    int v;
    printf("%sStudent %d assigned a slot on the Vaccination Zone %d and waiting to be vaccinated\n\n",GRN,id, i+1);
    if(students[id-1].round > 3)
    {
        printf("%sStudent %d on Vaccination Zone %d has been vaccinated 3 times now. He/She will be sent back to home!\n\n",MAG,id, i+1);
        v=1;
    }
    else
    {
        printf("%sStudent %d on Vaccination Zone %d has been vaccinated which has success probability %d%%\n\n",WHT,id, i+1,companies[zones[i].company_supply].prob);
        int x = rand()%101;
        if(x<companies[zones[i].company_supply].prob)
        {
            printf("%sStudent %d tested positive for antibody test. He/She can enter the college now!\n\n",WHT,id);
            v=1;
        }
        else
        {
            (students[id-1].round)++;
            printf("%sStudent %d tested negative for antibody test. Testing him/her again....\n\n",WHT,id);
            v=0;
        }
    }
    pthread_cond_signal(&(zones[i].condt_student));
    pthread_mutex_unlock(&(zones[i].zone_mutex));
    return v;
}
```

This function is called when a student needs to be vaccinated. If it is the 1st or 2nd or 3rd round of vaccination of a student, he/she is vaccinated and checked for antibodies. If antibodies are present, he/she has completed vaccination and can enter the college, else he/she is sent back to the queue at the gate and will be rechecked again. If it is >3rd round for a student, he/she is sent home. It returns 1 when a student is vaccinated and found positive for antibody test and 0 when he/she needs to be tested again.

4. student_waiting_slot

```
void student_waiting_slot(int id)
{
    bool zone_ready =false;

    printf("%sStudent %d is waiting to be allocated a slot on a Vaccination
Zone\n\n",YEL, id);

    while(zone_ready == false)
    {
        for(int j=0;j<M;j++)
        {
            pthread_mutex_lock(&(zones[j].zone_mutex));

            if(zones[j].slots > zones[j].strength)
            {
                zones[j].strength++;
                zone_ready = true;
                int k = student_vaccinated(j, id);
                if(k==0)
                {
                    if(zones[j].slots > zones[j].strength)
                    {
                        zones[j].strength++;
                        k = student_vaccinated(j, id);

                        if(k==0)
                        {
                            if(zones[j].slots > zones[j].strength)
                            {
                                zones[j].strength++;
                                k = student_vaccinated(j, id);
                                if(k==0)
                                    printf("%sStudent %d on Vaccination Zone %d has
been vaccinated 3 times now. He/She will be sent back to home!\n\n",MAG,id, j+1);

                            }
                        }
                    }
                }
                break;
            }

            pthread_cond_signal(&(zones[j].condt_student));
            pthread_mutex_unlock(&(zones[j].zone_mutex));
        }
    }
}
```

This function iterates over the zones and checks for the availability of slots while the loop is not exited. When a slot is available, it calls the **student_vaccinated** function and gets the value **1** if the student has entered the college or sent to home or **0** if the student has to be vaccinated again. If a student is vaccinated for 3 times and still failed antibody test, he/she will be sent home.

5. main

Takes input from the user and initializes all variables and mutexes. Also waits for all student threads to join. And finally, terminates the program.

Thread Functions

1. company_thread

In this function, the company prepares vaccines. Sleep is used for a random amount of time to stimulate the process of preparation of vaccines. As a result, a company **c** takes **w** seconds to produce **r** batches of vaccines, each with **p** vaccines and probability of success as **prob**. Then the **vaccine_ready** function is called for this particular company.

```
void * company_thread(void* args)
{
    Company * c = (Company *)args;

    while(1>0)
    {
        int r = rand()%10 + 1;
        int w = 2 + rand()%4;
        int p = rand()%11 + 10;
        printf("%sPharmaceutical Company %d is preparing %d batches of vaccines 
which have success probability %d%%.\n\n",CYN, c->id, r, c->prob);
        sleep(w);
        pthread_mutex_lock(&(c->company_mutex));
        c->batches = r;
        c->p = p;
        printf("%sPharmaceutical Company %d has prepared %d batches of vaccines 
which have success probability %d%%.\n\n",BLU, c->id, c->batches, c->prob);
        vaccine_ready(c);
    }

    return NULL;
}
```

2. zone_thread

In this function, we loop through all companies to check if vaccines are availables and since we use **pthread_mutex_lock** , only one zone at a time can access a company's information.

Once we find a company with vaccines, the zone decreases the value of vaccines held by the company, i.e., takes vaccines from the company and signals to conditional wait thread and waits. Else it continues to check for availability of vaccines.

```
    for(int i=0;i<N;i++)
     {
     pthread_mutex_lock(&(companies[i].company_mutex));
         if(companies[i].batches != 0)
         {
             flag = true;
             companies[i].batches--;
             z->p = companies[i].p;
             z->company_supply = i;
             printf("%sPharmaceutical Company %d is delivering vaccines to
Vaccination zone %d which has success probability %d%%.\n\n%sPharmaceutical
Company %d has delivered vaccines to Vaccination zone %d, resuming
vaccinations now.\n\n",CYN,i+1,z->id,companies[z->company_supply].prob,
BLU,i+1,z->id);

             pthread_cond_signal(&(companies[i].condt_zone));
             pthread_mutex_unlock(&(companies[i].company_mutex));
             break;
         }
         sleep(0.01);
         pthread_cond_signal(&(companies[i].condt_zone));
         pthread_mutex_unlock(&(companies[i].company_mutex));
     }
```

Once a company is found, it locks its mutex and checks for vaccines. If a company has run out of vaccines, it unlocks its mutex. After acquiring the vaccines, zones generate **k** number of slots where k>=1 and k<=minimum(8, number of vaccines left in the batch, number of students waiting). Then **ready_vaccination_zone** function is called.

```
     pthread_mutex_lock(&(z->zone_mutex));
         if(z->p == 0 && flag == true)
         {
             printf("%sVaccination Zone %d has run out of vaccines\n\n",
MAG, z->id);
             pthread_mutex_unlock(&(z->zone_mutex));
             break;
         }
```

```
        z->strength = 0;
        int k = rand()%min(z->p,min(7,O))+1;
        z->slots = k;
        z->slots = min(z->slots,z->p);
        z->p-=z->slots;
        printf("%sVaccination Zone %d is ready to vaccinate with %d
slots.\n\nVaccination Zone %d entering Vaccination Phase.\n\n",
YEL,z->id,z->slots,z->id);
        ready_vaccination_zone(z);
```

3. student_thread

Initially, it sleeps for some random time to stimulate the delay in arrival of students. Then once a student arrives, it calls **student_waiting_slot** function for that particular student.

```
void * student_thread(void* args)
{
    Student * s = (Student*)args;
    sleep(rand()%4); // to stimulate a delay in students arriving
    printf("%sStudent %d has arrived for his 1st round of
Vaccination\n\n",YEL,s->id);
    student_waiting_slot(s->id);
    return NULL;
}
```