Computer Systems Engineering-1

Assignment 4

Tanish Lad - 2018114005

Automatic Biryani Serving Question Report

OVERVIEW

The code is an attempt to simulate an Automatic Biryani Serving System which consists of N Chefs, M Tables and K Students.

Functions Used

1. biryani_ready:

   It waits for all the vessels prepared by the chef, specified by the id in the function parameter, to get emptied.

```
while(chefs[id-1].noOfVessels > 0 && K > 0)
{
    // do nothing;
}
```

2. prepareBiryani:

   It is the function called by each thread, a thread signifying once chef, and as long as there are some students left to eat biryani, it goes on making a random number of vessels of biryani having a random capacity in a random amount of time.

```
while(K > 0)
{
    srand(time(0));
    int w = 2 + rand()%4;
    int noOfVessels = rand()%10 + 1;
    int serves = 24 + rand()%26 + 1;

    if(K > 0)
    {
        printf("Robot Chef %d is preparing %d vessels of Biryani\n", c -> id, noOfVessels);
        // printf("Chef %d has prepared %d vessels of biryani each having a capacity to serve %d student
        // fflush(stdout);
    }

    sleep(w);

    pthread_mutex_lock(&(c -> lockChef));

    c -> noOfVessels = noOfVessels;

    c -> p = serves;

    printf("Robot Chef %d has prepared %d vessels of Biryani. Waiting for all the vessels to be emptied
    c -> prepared = 1;
```

3. ready_to_serve_table:

```
while(tables[id-1].servedTillNow < tables[id-1].slotsCopy && K > 0)
{
    // printf("*** %d ** %d ***\n", t -> noOfSlots, t -> id);

    // do nothing;
}
```

It waits until all the students, the table (specified by the id in the function parameter) is currently serving aren't served.

4. serveBiryani

This function is called by each thread, a thread specifying one table, and until and unless there are some students left to be served biryani, the table finds a chef which has prepared

biryani, and if it finds one, it takes a vessel of biryani from the chef and starts serving students.

```
while(t -> found == 0 && K > 0)
{
    // printf("*\n");
    for(int i = 0; i < N; i++)
    {
        // printf("%d * %d\n", t -> id, i);
        pthread_mutex_lock(&(chefs[i].lockChef));
        pthread_mutex_lock(&(t -> lockTable));

        if((chefs[i].prepared) == 1)
        {
            // if(chefs[i].noOfVessels > 0)
            // {
                t -> p = chefs[i].p;
                // (t -> refilled)++;

                // if(K > 0)
                // {
                    printf("Robot Chef %d is refilling Serving Container

                    // if(t -> refilled > 1)
                    // {
                        printf("Serving Container of Table %d is refilled
                    // }

                    // printf("Serving Table %d is ready to serve with %d
                    // printf("Table %d is ready to serve biryani with ca
                    // fflush(stdout);
                // }

                (chefs[i].noOfVessels)--;
            // }
```

5. student_in_slot

   This function signifies that a student has got a slot on a table and is waiting for the table to serve him/her biryani.

   After the table serves him biryani, the function signifies that by printing that he/she has been served biryani.

6. wait_for_slot

```
printf("Student %d is waiting to be allocated a slot on the serving table\n", s -> id);
while(s -> found == 0)
{
    for(int i = 0; i < M; i++)
    {
        pthread_mutex_lock(&(tables[i].lockTable));
        if(tables[i].noOfSlots > 0)
        {

            // if(tables[i].noOfSlots > 0)
            // {
                s -> tableID = tables[i].id;
                printf("Student %d assigned a slot on the serving table %d and waiting
                // printf("Student %d is waiting to be served at table %d\n", s -> id,

                (tables[i].noOfSlots)--;
                // printf("\t\t\t* at student %d * and table %d * slots %d * %d *\n", s
            // }

            // else
            // {
            //     pthread_mutex_unlock(&(tables[i].lockTable));
            //     continue;
            // }

            // // fflush(stdout);

            pthread_mutex_lock(&lockStudentsLeft);
            K--;
            pthread_mutex_unlock(&lockStudentsLeft);

            s -> found = 1;
            pthread_mutex_unlock(&(tables[i].lockTable));
```

This function is called by each thread, a thread specifying one student, and it iterates on all the tables until it finds a table which is ready to serve him/her biryani, and if it finds one, it goes to that table and waits for the table to serve him/her biryani.