# Automated biryani serving system

## Basic structure and pipeline

there are r robot cheffs , t tables and n students who want to eat in the mess

every robot is a thread who prepares biryani and waits for any table thread to take the biryani from the robot chef

every table is also who a thread which iterates through the robot array waiting for some vessel to be prepared which is then loaded on to the table

once a vessel is loaded table thread creats random number of slots waits for the students to arrive at those slots . once all the slots are full , the students eat and then if there is still biryani left in the container the table repeats the above .

every student is also a thread who arrives at a random time in the m,ess and iterates through the table looking for empty slots . once an empty slot is found , the student eats

## Robot Chef

- Each Robot chef takes **w seconds** (random between 2-5) to prepare **r vessels** (random between 1-10) of biryani at any particular time. Each vessel has a capacity to feed **p students** (random between 25-50).

```
void * chef (void *inp)
{
    struct chef * me = (struct chef *) inp;
    printf("chef %d arrived \n",me->chefid);
    while (1)
    {
        int w =2+ rand()%3;
        int r =1+ rand()%10;
        int p =25+ rand()%25;
        me->waittime=w;
        sleep(w);

        pthread_mutex_lock(&me->cheflock);
```

```
        me->vesselrem=r;
        me->vesselcap=p;

        printf(ANSI_COLOR_RED"chef %d prepared %d vessels of %d capacity each
 \n"ANSI_COLOR_RESET,
            me->chefid ,me->vesselrem ,me->vesselcap );
        biryani_ready( me );

    }
    return NULL;
}
```

- Once the Robot Chef is done preparing the Biryani Vessels, he invokes the function biryani_ready() and does not return from it until all the biryani vessels he has cooked are loaded into any of the serving containers.

the biryani_ready() function waits for some table to pick up the prepared biryani vessel and returns once all the vessels are picked .

```
void biryani_ready(struct chef * currchef)
{
    while(1)
    {
        if(currchef->vesselrem<=0)
        {
            printf(ANSI_COLOR_RED "All the vessels prepared by Robot Chef %d ar
e emptied. Resuming cooking
                    now. \n"ANSI_COLOR_RESET,currchef->chefid );
            break;
        }
        else
        {
            pthread_cond_wait(&currchef->chefcond,&currchef->cheflock);
        }
    }
    pthread_mutex_unlock(&currchef->cheflock);
}
```

- Once all the biryani vessels are empty, the biryani_ready() function returns and the Robot Chef resumes making another batch of Biryani.

# Serving Tables

● Each Serving table’s serving container is initially empty. it iterates through all the chefs to look for a prepared vessel to load into container .

```
    printf(ANSI_COLOR_GREEN " Table %d Waiting for Biryani \n" ANSI_COLOR_RESE
T,self->tableid );
    while (1)
    {
       itr++;
       itr = itr%rcheffno;
       struct chef * ichef = cheffarr[itr];
       pthread_mutex_lock(&ichef->cheflock);
       if(ichef->vesselrem>0)
       {
           self->containers=ichef->vesselcap;
           ichef->vesselrem--;
           pthread_cond_signal(&ichef->chefcond);
           pthread_mutex_unlock(&ichef->cheflock);
           break;
       }
       else
           pthread_mutex_unlock(&ichef->cheflock);
    }

    printf(ANSI_COLOR_GREEN " Table %d got Biryani from chef %d \n" ANSI_COLOR_
RESET,self->tableid,itr );
```

● Once the serving container is full, it enters into the serving mode. It invokes a function ready_to_serve_table(int number_of_slots) in which number_of_slots (chosen randomly between 1-10) denotes the number of slots available at the particular serving table at that instant. If there is no waiting student, then the function returns.

```
void ready_to_serve(struct table * tabol)
{
    while (1)
    {
        printf("slots remaining on %d are %d\n",tabol->tableid,tabol->slotsre
m);
        if (tabol->slotsrem<=0)
            break;
        else
        {
            pthread_cond_wait(&tabol->tablecond,&tabol->tablelock);
        }
    }
    pthread_mutex_unlock(&tabol->tablelock);
}
```

# Student

● Once a student arrives in the mess, he/she invokes a function wait_for_slot(). Once the function returns the students goes to the allocated slot and waits for the slot to serve Biryani.

● upon arriving the student iterates through the tables looking for slots and assigns itlself whichever table has a vacant slot

```
void * students (void * inp )
{
    struct student * thisisme = (struct student * )inp;
    int waittimeba = rand()%10;

    sleep(waittimeba);
    printf("STUDENT %d ARRIVED IN THE MESS \n",thisisme->stuid);
    wait_for_slot(thisisme);
```

```
    return NULL;
}
```

- function waiting for slots

```
void wait_for_slot(struct student * thisisme)
{
    int i=0;
    int found =0;
    while (found ==0 )
    {
        i++;
        i%=tableno;
        struct table * itrtable = tablearr[i];
        pthread_mutex_lock(&itrtable->tablelock);

        if (itrtable->slotsrem > 0 )
        {
            itrtable->slotsrem--;
            found ++;
            pthread_cond_signal (&itrtable->tablecond);
            printf(ANSI_COLOR_YELLOW"student %d assigned a slot on table %d\n"A
NSI_COLOR_RESET,thisisme->stuid ,i);
        }

        pthread_mutex_unlock(&itrtable->tablelock);
    }
}
```

# Assumption

- a student will be served whenever he is assigned a slot on a table