

# Operating Systems-2

## Report-Assignment4

### Theory & Programming Assignment 4: The Korean Restaurant Problem

Name: Naitik Malav

Roll No.: CS19BTECH11026

1. First of all I have defined a class namely *Info* which have the parameters  $n$ ,  $x$ ,  $r$ ,  $\lambda$ ,  $\tau$ . It stores the data from file *input.txt* into object *information*.

```
class Info {
public:
    int n; //number of people
    int x; //number of seats
    double lambda;
    double r;
    double tau;

    void getParameter();
} information;
```

About function **getParameter** : Assign all parameters from the input file to class members.

2. *Thread* is a vector of threads. Here I have passed *Korean* function to each thread along with the thread id and pushing it to Thread vector.

Using below technique I have passed  $s$  set of customers using `Thread.push_back`.

```
while(track < information.n) { //until track less than total number of people
    distribution1 = new exponential_distribution<double>(information.lambda);
    usleep((*(distribution1)(Generator)*1000000);

    for(int i=track; i<(int)s; i++) {
        Threads.push_back(thread(Korean, i)); //creating thread
    }

    for(int i=track; i<(int)s; i++) {
        Threads[i].join(); //joining threads
    }
    track = (int)s;

    while(track+int(s)>information.n && track<information.n) {
        s = (double)1 + (double)(rand()) / ((double)(RAND_MAX/((double)information.r*(double)information.x - (double)1)));
    }
    s = (double)track + (double)1 + (double)(rand()) / ((double)(RAND_MAX/((double)information.r*(double)information.x - (double)1)))
}
```

About function **Korean** : It takes thread id as parameter. And we will pass this function along with the thread id to each thread and pushing all the required time like request access time, given access time and left time into vector namely *Time*. About the algorithm part of Korean it is according to the pseudocode provided.

3. *exponential\_distribution* is used to generate random number which is basically a floating point value and it described by probability function:

$$P(x|k) = ke^{-kx}$$

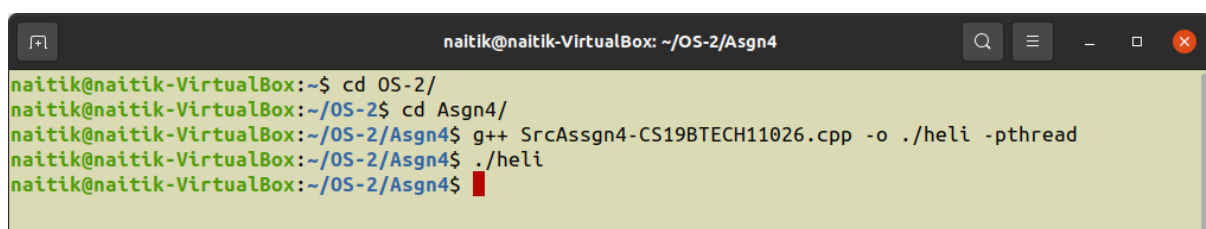
*distribution1* and *distribution2* are 2 exponential distributions defined globally which is used for time delay of eating and of entering into the restaurant.

5. *usleep* is used to suspend the execution of a thread by few microseconds. So, I have multiplied with 10us inside *usleep* so that it won't run for longer time neither for shorter time.

6. Using buffer vector<string>Time, in which I push down all the time along with the statements and printing it at last in Log-CS19BTECH11026.txt file.

7. *getSysTime* is used to record the system time.

8. Compilation Screenshot:



```
naitik@naitik-VirtualBox: ~/OS-2/Asgn4
naitik@naitik-VirtualBox:~$ cd OS-2/
naitik@naitik-VirtualBox:~/OS-2$ cd Asgn4/
naitik@naitik-VirtualBox:~/OS-2/Asgn4$ g++ SrcAsgn4-CS19BTECH11026.cpp -o ./heli -pthread
naitik@naitik-VirtualBox:~/OS-2/Asgn4$ ./heli
naitik@naitik-VirtualBox:~/OS-2/Asgn4$
```

9. For input parameter:

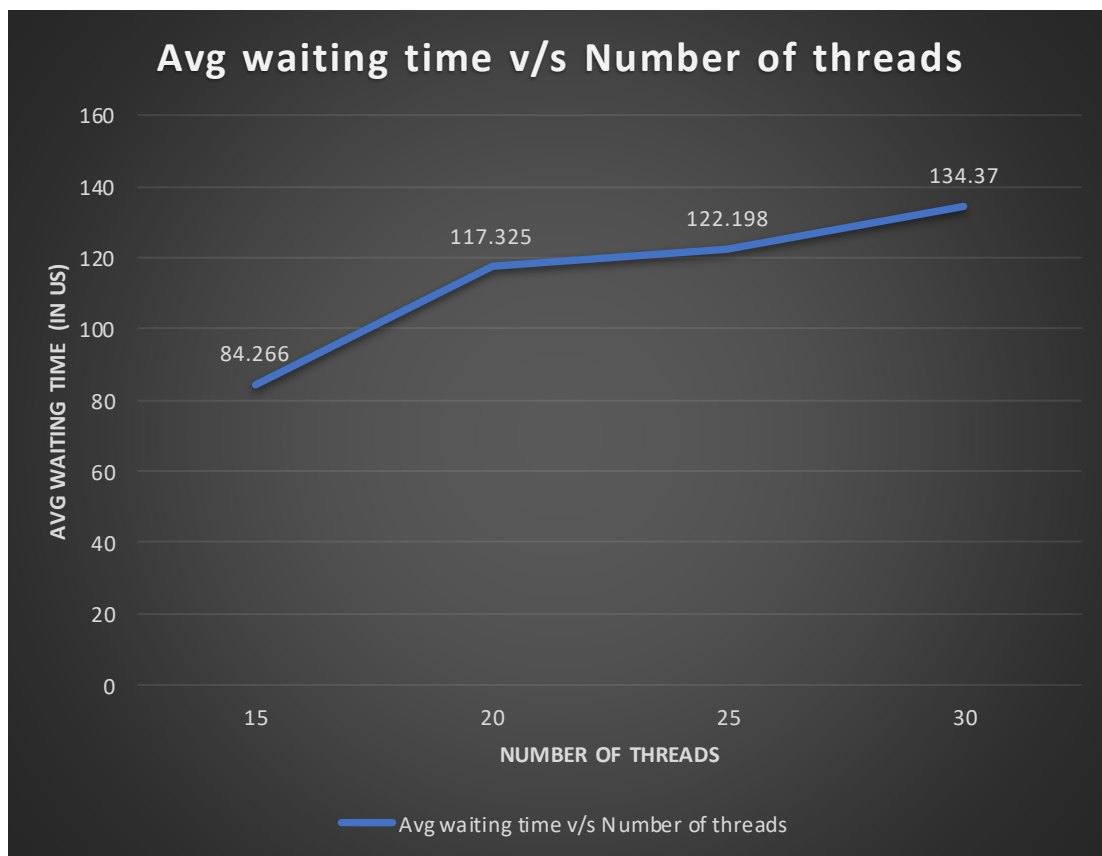
$$n=7, x=4, \text{lambda}=0.7, r=0.8, \text{tau}=0.9$$

Log outputs are as follows:

```
SrcAssgn4-CS19BTECH11026.cpp  input.txt  Log-CS19BTECH11026.txt X
Asgn4 > Log-CS19BTECH11026.txt
1 1th customer access request at time: 20:58:32
2 1th given access at time: 20:58:32
3 2th customer access request at time: 20:58:32
4 2th given access at time: 20:58:32
5 2th left at time: 20:58:33
6 1th left at time: 20:58:36
7 3th customer access request at time: 20:58:36
8 3th given access at time: 20:58:36
9 3th left at time: 20:58:39
10 4th customer access request at time: 20:58:40
11 4th given access at time: 20:58:40
12 5th customer access request at time: 20:58:40
13 5th given access at time: 20:58:40
14 5th left at time: 20:58:41
15 4th left at time: 20:58:46
16
```

**Graph1:**

Number of Threads	Average Waiting time(in us)
15	84.266
20	117.325
25	122.198
30	134.37



**Graph2:**

Value of X	Avg Waiting time(in us)
4	139.11
5	166.465
6	184.9832
7	222.09974
8	231.198

