Ober Cab Services

Basic Structure and piping

- Each cab is just a struct which stores the current status of the cab and the id of the cab
- there is an array of cabs which stores the pointer to the struct of every cab at an index which equals the id of the cab

```
struct cab {
    int cabid ;
    int cabstatus ;
    int inpassengerid;
    int inpassengerid2;
};
```

Passenger arrival and cab search

- every passenger is a thread and as soon as a passenger arrives after a random wait time, it randomly chooses a particular cab type (either pool or premier) and looks for the availability of that cab type .

- If the cab type is not available at that moment, the passenger waits until its max wait time .

- **There is NO Busy waiting** and pthread_cond_timed wait() is used to make the passenger wait

Leaving a cab

- whenever a passenger wants to leave the cab , it iterates through all the

passengers and checks if there is any passenger waiting for the cab that he wants to leave.

- if no such waiting passenger is found , the count of available cabs of the respective type is increased for the passengers yet to arrive

Payment section

- upon leaving a cab every passenger signals the payment servers for payment by posting to a semaphore which basically keeps the count of total number of passengers who want to pay
- every server waits on that semaphore and once signalled , it looks for the passenger who requested the payment and processes the payment .
- semaohore was used to avoid deadlocks and busy waiting

code segments and important functions

passenger initialisation and cab type request

```
    int willwait =4;

    struct passenger* args  = (struct passenger*) passinp;
    int k= rand()%maxarrivaltime;
    int ctype = rand()%2;
    if (ctype==0)
        args->cabtype=0;
    else
        args->cabtype=1;
    sleep(k);
    if (ctype==0)
        printf("\033[1;31m passenger %d has arrived and wants a pool ca
b\n\033[0m"
```

```
                ,args->passid);
    else
      printf("\033[1;31mpassenger %d has arrived and wants a premier ca
b\n\033[0m"
                ,args->passid);

    struct ridedata rd;
    rd.passdata=args;
    pthread_mutex_lock(&(pass_array_lock));
```

looking for cab

similar code segments are used to look for pool cabs

```
printf("wanted premier %d\n",args->passid);
        if (empty>0)
        {
            //got empty
            printf("passenger %d got an empty ",args->passid);
            for (int i=0;i<cno;i++)
            {
                if (cabarray[i]->cabstatus == 0)
                {
                    printf("with cabid %d \n",i);
                    cabarray[i]->cabstatus = 3;
                    cabarray[i]->inpassengerid=args->passid;
                    empty--;
                    rd.cabid=i;
                    break;
                }
            }
            pthread_mutex_unlock(&(pass_array_lock));
```

```
//            printf("\n\ncabid 4%d \n\n",rd.cabid);
            enjoyride(rd);
        }
```

waiting for cab

```
            // did not get empty
            passenarray[args->passid]=2;
            int timeflag = pthread_cond_timedwait(&(condarray[args->pas
sid]),&(pass_array_lock),gettime(willwait));
```

- Function enjoy_ride() is used to simulate the entire ride of the passenger and also leave the cab and assign is to some other passenger
- it also sends the request for payment

leaving a cab

```
        {
            if (cabdata->cabstatus == 3 )
            {
                assigned ++;
                passenarray[i]=0;
                cabdata->cabstatus=0;
                pthread_cond_signal (&(condarray[i]));
                pthread_mutex_unlock(&pass_array_lock);
                break;
            }
            if (cabdata->cabstatus == 1 )
            {
                cabdata->cabstatus=0;
                p1--;
                assigned ++;
                passenarray[i]=0;
```

```
                pthread_cond_signal (&(condarray[i]));
                pthread_mutex_unlock(&pass_array_lock);
                break;
            }
```

server thread

```
void * server (void * inp)
{
    struct server * self = (struct server * )inp;
    while (1)
    {
        int found =0 ;
        sem_wait(&payment_semaphore);
        pthread_mutex_lock(&servermutex);
        int plol;
        for (int i=0;i<pno;i++)
        {
            if (paymentarr[i]==1)
            {
                plol=i;
                found++;
                printf("\033[1;36m passenger %d is paying on server %d
\n\033[0m",
                        i,self->serverid);
                paymentarr[i]=0;
            }
        }
        pthread_mutex_unlock(&servermutex);
        if (found !=0)
        {
            sleep(2);
            printf("\033[1;36m passenger %d payment completed\n \033[0
```

```
m",
                    plol);
        }
        if (found==0)
        {
            break;
        }

    }
    return NULL;
}
```