**OS Assignment**

**Lab – 11**

**RollNo-012**

**Aim**: Study and implementation of Sleeping barber and Dining philosopher situations .

**Code-1**: **Sleeping Barber**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>

#define TOTAL\_CUSTOMERS\_COUNT 1000

#define BARBERS\_COUNT 2

#define WAITING\_CAPACITY 5

sem\_t sem\_waitCount,sem\_customers,sem\_barbers;

int waitingCustomerCount=0;

void \*Customer(void \*arg){

int \* custId=(int \*)arg;

sem\_wait(&sem\_waitCount);

if(waitingCustomerCount<WAITING\_CAPACITY){

waitingCustomerCount++;

printf("Customer-%d : entered waiting area\n",\*custId);

sem\_post(&sem\_customers);

sem\_post(&sem\_waitCount);

sem\_wait(&sem\_barbers);

printf("Customer-%d : got haircut exiting\n",\*custId);

}

else{

sem\_post(&sem\_waitCount);

}

}

void \*Barber(void \*arg){

int waitingTemp=0;//,cust\_id=-1;

while(1){

sem\_wait(&sem\_customers);/\*wait if no customers present\*/

/\*when there is customer\*/

sem\_wait(&sem\_waitCount);/\*select one customer and decrement the waiting count\*/

waitingCustomerCount--;

waitingTemp=waitingCustomerCount;

printf("\tBarber: started cutting hair\ttotal tcustomers in waiting area: %d \n",waitingTemp);

sem\_post(&sem\_barbers);

sem\_post(&sem\_waitCount);

}

}

void main(){

int count1,cust\_id[TOTAL\_CUSTOMERS\_COUNT];

pthread\_t barber\_array[BARBERS\_COUNT],customer\_array[TOTAL\_CUSTOMERS\_COUNT];

printf("threads created\n");

sem\_init(&sem\_barbers,0,0);

sem\_init(&sem\_customers,0,0);

sem\_init(&sem\_waitCount,0,1);

for(count1=0;count1<BARBERS\_COUNT;count1++){

pthread\_create(&barber\_array[count1],NULL,Barber,(void\*)NULL);

}

for(count1=0;count1<TOTAL\_CUSTOMERS\_COUNT;count1++){

cust\_id[count1]=count1+1;

pthread\_create(&customer\_array[count1],NULL,Customer,(void\*)&cust\_id[count1]);

}

for(count1=0;count1<BARBERS\_COUNT;count1++){

pthread\_join(barber\_array[count1],NULL);

}

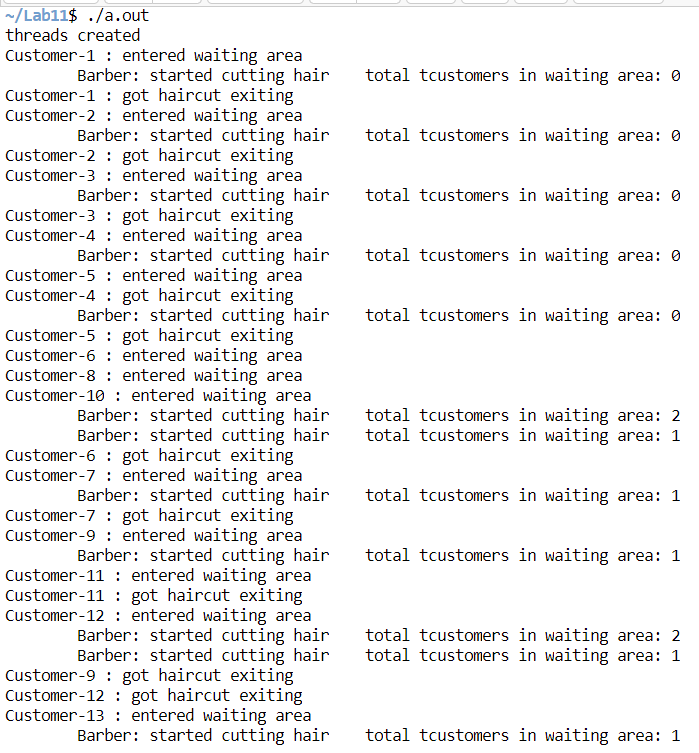
for(count1=0;count1<TOTAL\_CUSTOMERS\_COUNT;count1++){

pthread\_join(customer\_array[count1],NULL);

}

}

**O/P**:



**Code-2:** writer have priority

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>

#define P\_COUNT 5

sem\_t s\_fork\_set[P\_COUNT],s\_room;

void \*Philosopher(void \*arg){

int \*phId=(int \*)arg;

while(1){

sem\_wait(&s\_room);

printf("P-%d is thinking\n",\*phId);

sem\_wait(&s\_fork\_set[\*phId-1]);

printf("\t\tP-%d : grabbed left fork\n",\*phId);

sem\_wait(&s\_fork\_set[(\*phId)%P\_COUNT]);

printf("\tP-%d : grabed both fork and is ready to eat\n",\*phId);

sem\_post(&s\_fork\_set[(\*phId)%P\_COUNT]);

printf("\t\tP-%d : relesed right\n",\*phId);

sem\_post(&s\_fork\_set[\*phId-1]);

printf("\t\tP-%d : relesed both fork\n",\*phId);

sem\_post(&s\_room);

sleep(1);

}

}

void main(){

pthread\_t t\_philosophers[P\_COUNT];

int count1,ph\_id[P\_COUNT];

sem\_init(&s\_room,0,P\_COUNT-1);

for(count1=0;count1<P\_COUNT;count1++){

sem\_init(&s\_fork\_set[count1],0,1);

}

printf("semaphores are ready\n");

//run

for(count1=0;count1<P\_COUNT;count1++){

ph\_id[count1]=count1+1;

pthread\_create(&t\_philosophers[count1],NULL,Philosopher,(void\*)&ph\_id[count1]);

}

//join

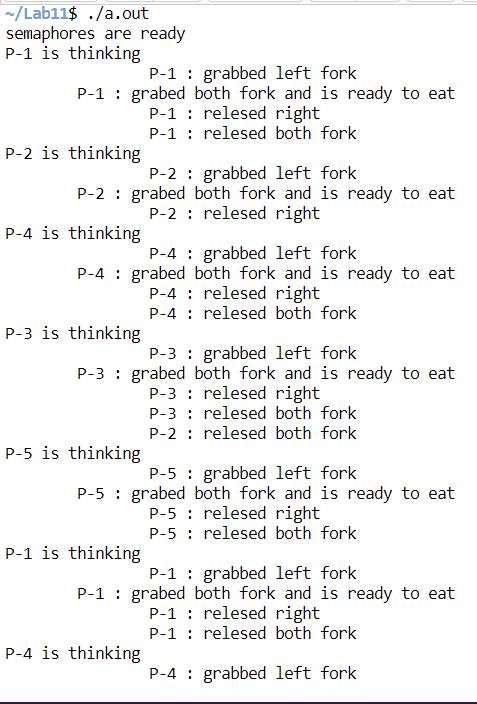
for(count1=0;count1<P\_COUNT;count1++){

pthread\_join(t\_philosophers[count1],NULL);

}

}

**O/P**:



* **Some Exceptional conditions or errors**:
  + It may happen that threads are scheduled biasedly/unevenly by the kernel sometimes