

## **Practica 2 - Sistemas operativos**

Alejandro Fernandez Restrepo  
201910007010

Profesora:

Carmen Liliana Carvajal Jimenez

ingeniería de sistemas

**Eafit**  
**2021-2**

### Instructions:

**You must submit the solution of the following problem on or before October 6 2021.**

**You must submit: 1. The source code, 2. A pdf report with the source code, 3. A short video explaining (with your voice) the output according to the source code.**

(The Sleeping Barber Problem [Dijkstra, 1965]) A barbershop consists of a waiting room with 10 chairs and a barber room containing the barber chair. If there are no customers to be served, the barber goes to sleep. If a customer enters the barbershop and finds the barber asleep, he wakes him up.

Write a deadlock-free program to coordinate the barber and the customers. Your code must be based on the producer-consumer discussed in class. The program must read the number of chairs and it must generate a customer randomly, the client arriving time must be between 0 and 3 seconds. Each customer arrives asking for a specific cut hair style of the following:

Cut hair	Lasting time
1	500ms
2	1000ms
3	2000ms
4	3000ms

The cut hair style must be generated randomly for each client.

The program must indicate:

- When a client arrives and the number of empty chairs and the cut hair style for each client in the queue
- When the barber pick-up a client and the style asked by the client.
- When the barber drop a client and the time consumed by the client.

## 1.Ejecucion

Descargar el código fuente desde el repositorio o en su defecto descargando el código desde la entrega en interactiva.

Antes de poder correr el código se tiene que verificar que se tengan las herramientas apropiadas para su ejecución :

-Un editor de texto

-Tener C instalado en nuestra maquina

O en su defecto usar un ejecutor de código online como lo es replit el cual usare para explicar y ejecutar el código.

Para ejecutar el programa debemos usar el comando:

- **gcc main.c -o main**

y después ./main O en su defecto darle run en replit

## 2.Solucion

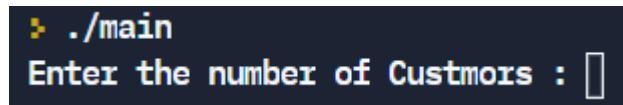
Se usaron 4 semaforos para poder solucionar el problema, los cuales son:

-Semáforo waiting room : Limita el numero de clientes que pueden entrar a la barberia ( el numero de sillas).

-semaforo barberchair : Representa la silla del barbero donde los clientes pasan a ser atendidos.

-Semaforo barberSleep: permite al barbero dormir hasta que un cliente llegue y lo despierte

-Semaforo haircut: hace esperar al cliente hasta que su corte de pelo este listo.



```
➤ ./main
Enter the number of Customers : █
```

Cuando se ejecuta el programa lo primero que se hace es que se pide el numero de clientes que deseamos que vayan a la barberia.

des pues de esto se instancian todos los semáforos usando sem\_init, se crea el thread del barbero y los thread de los clientes, se espera a que todos los thread de los clientes terminen y al final se termina el thread del barbero

```
// Initialize the semaphores
sem_init(&waitingRoom, 0, numChairs);
sem_init(&barberChair, 0, 1);
sem_init(&barberSleep, 0, 0);
sem_init(&hairCut, 0, 0);

// Create the barber.
pthread_create(&btid, NULL, barber, NULL);

// Create the customers.
for (i=0; i<numCustomers; i++) {
    pthread_create(&tid[i], NULL, customer, (void *)&Number[i]);
    sleep(1);
}

// Join each of the threads to wait for them to finish.
for (i=0; i<numCustomers; i++) {
    pthread_join(tid[i], NULL);
    sleep(1);
}

// When all of the customers are finished, finish the barber
allDone = 1;
sem_post(&barberSleep); // Wake the barber so he will exit.
pthread_join(btid, NULL);
return 0;
```

## Funcionamiento del barbero

El barbero siempre estará activo hasta que no halla clientes, el barbero se ira a casa en caso tal de que no halla mas clientes. El barbero solo se puede ir a dormir si no hay nadie en la sala de espera(en caso tal de que este dormido el cliente lo despierta usando `sem_post(&barberSleep)`).

El barbero siempre le va a preguntar al cliente que tipo de corte de pelo quiere y dependiendo de este se esperara un tiempo de espera concreto , el cual se describe en la función(`cuttime`).

Cuando el barbero termina de cortar el pelo este deja ir al cliente a través de (`sem_post(&hairCut)`)

```
while (!allDone) {

    int customers;
    // If there are no customers to be served, the barber goes to sleep
    sem_getvalue(&waitingRoom,&customers);
    if(customers==10){
        printf("The barber is sleeping \n");
        sem_wait(&barberSleep);
    }

    if (!allDone) {
        // customer's hair.
        printf("The barber is asking the hair cut style\n");
        //tThe cut hair style must be generated randomly for each client.
        cuttime(rand() %4);
        printf("The barber has finished cutting hair.\n");
        // Release the customer when done cutting
        sem_post(&hairCut);
    }
    else {
        printf("The barber is going home for the day.\n");
    }
}
```

## Funcionamiento del cliente

Cada cliente se demorara aleatoriamente entre 0 y 3 segundos en llegar a la barberia , cuando este llega mira si hay algún puesto libre para esperar a través de `sem_trywait(&waitingRoom)` , en caso tal de que no lo halla se va, este siempre va a decir cuantas sillas hay disponibles dentro de la barberia.

Después de esto espera a que la silla del barbero se de desocupe para poder pasar a ser atendido, si el barbero esta dormido este lo despierta usando `sem_post(&barberSleep)` después pasa a ser motilado , cuando termina este deja libre la silla del barbero y se va de la barberia

```
int num = *(int *)number;
int chairs;
int wake;
// Leave for the shop and take some random amount of
printf("Customer %d is going to the barber shop.\n", num);
// random time to arrive (0 - 3 seg).
sleep(rand()%3);
printf("Customer %d arrived at barber shop.\n", num);

// Wait for space to open up in the waiting room
if (sem_trywait(&waitingRoom) == -1) {
    printf("Waiting room full. Customer %d is leaving.\n", num);
    return 0;
}
sem_getvalue(&waitingRoom, &chairs);
printf("The number of empty chairs is %d\n", chairs);
printf("Customer %d entering waiting room.\n", num);

// Wait for the barber chair to become free.
sem_wait(&barberChair);

// The chair is free so give up your spot in the
// waiting room.
sem_post(&waitingRoom);

// Wake up the barber if it is sleeping
sem_getvalue(&barberSleep, &wake);
if(wake==0){
    printf("Customer %d waking the barber.\n", num);
    sem_post(&barberSleep);
}

// Wait for the barber to finish cutting your hair.
sem_wait(&hairCut);

// Give up the chair.
sem_post(&barberChair);
printf("Customer %d leaving barber shop.\n", num);
```

### 3. Código fuente

El código puede ser encontrado en el siguiente repositorio :

<https://github.com/afernander/SleepingBarberProblem>

O dentro de este pdf

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <time.h>
#include <pthread.h>
#include <semaphore.h>

// The maximum number of customer threads.
#define MAX_CUSTOMERS 25
//Number of chairs
#define NUM_CHAIRS 10

// Function prototypes...
void *customer(void *num);
void *barber(void *);

void cuttime(int num);
void timearrive(int time);

// Define the semaphores.

sem_t waitingRoom;
sem_t barberChair;
sem_t barberSleep;
sem_t hairCut;

int allDone = 0;

int main(int argc, char *argv[]) {
    pthread_t btid;
    pthread_t tid[MAX_CUSTOMERS];
    long RandSeed;
    int i, numCustomers, numChairs;
    int Number[MAX_CUSTOMERS];

    printf("Enter the number of Custmors : "); scanf("%d",&numCustomers);

    numChairs = NUM_CHAIRS;
```

```

// Make sure the number of threads is less than the number of customers
if (numCustomers > MAX_CUSTOMERS) {
printf("The max number of Customers is %d.\n", MAX_CUSTOMERS);
exit(-1);
}

// Initialize the numbers array.
for (i=0; i<MAX_CUSTOMERS; i++) {
Number[i] = i;
}

// Initialize the semaphores
sem_init(&waitingRoom, 0, numChairs);
sem_init(&barberChair, 0, 1);
sem_init(&barberSleep, 0, 0);
sem_init(&hairCut, 0, 0);

// Create the barber.
pthread_create(&btid, NULL, barber, NULL);

// Create the customers.
for (i=0; i<numCustomers; i++) {
pthread_create(&tid[i], NULL, customer, (void *)&Number[i]);
sleep(1);
}

// Join each of the threads to wait for them to finish.
for (i=0; i<numCustomers; i++) {
pthread_join(tid[i],NULL);
sleep(1);
}

// When all of the customers are finished, finish the barber
allDone = 1;
sem_post(&barberSleep); // Wake the barber
pthread_join(btid,NULL);
return 0;
}

void *barber(void *junk) {
// While there are still customers to be serviced the barber will be in
the barber shop
while (!allDone) {

    int customers;

```

```

// If there are no customers to be served, the barber goes to sleep
sem_getvalue(&waitingRoom,&customers);
if(customers==10){
    printf("The barber is sleeping \n");
    sem_wait(&barberSleep);
}

if (!allDone) {
    // customer's hair.
    printf("The barber is asking the hair cut style\n");
    //The cut hair style must be generated randomly for each client.
    cuttime(rand() %4);
    printf("The barber has finished cutting hair.\n");
    // Release the customer when done cutting
    sem_post(&hairCut);
}
else {
    printf("The barber is going home for the day.\n");
}
}
}

void *customer(void *number) {
    int num = *(int *)number;
    int chairs;
    int wake;
    // Leave for the shop and take some random amount of
    printf("Customer %d is going to the barber shop.\n", num);
    // random time to arrive (0 - 3 seg).
    sleep(rand()%3);
    printf("Customer %d arrived at barber shop.\n", num);

    // Wait for space to open up in the waiting room
    if (sem_trywait(&waitingRoom) == -1) {
        printf("Waiting room is full. Customer %d is leaving.\n",num);
        return 0;
    }
    sem_getvalue(&waitingRoom,&chairs);
    printf("The number of empty chairs is %d\n",chairs);
    printf("Customer %d entering waiting room.\n", num);

    // Wait for the barber chair to become free.
    sem_wait(&barberChair);

    // The chair is free so give up your spot in the

```



```

// waiting room.
sem_post(&waitingRoom);

// Wake up the barber if it is sleeping
sem_getvalue(&barberSleep,&wake);
if(wake==0){
    printf("Customer %d waking the barber.\n", num);
    sem_post(&barberSleep);
}

// Wait for the barber to finish cutting your hair.
sem_wait(&hairCut);

// Give up the chair.
sem_post(&barberChair);
printf("Customer %d leaving barber shop.\n", num);
}

//cutting hair time for each type there is a diferent waitng time
void cuttime(int num){
    printf("The customer want the haircut number %d\n",num);
    if (num==1){
        printf("The barber is cutting hair\n Waiting time is 500ms\n");
        sleep(500/1000);

    }else if (num==2){
        printf("The barber is cutting hair\n Waiting time is 1000ms\n");
        sleep(1);
    }else if (num==3){
        printf("The barber is cutting hair\n Waiting time is 2000ms\n");
        sleep(2);
    }else if (num==4){
        printf("The barber is cutting hair\n Waiting time is 3000ms\n");
        sleep(3);
    }
}
}
}
}

```