Barbero Durmiente

Descripción

Un Barbero que atiende clientes. Al terminar de atender un cliente, el barbero comprueba la sala de espera, y si hay algún cliente esperando lo atiende. Si no hay clientes en la sala de espera, se pone a dormir. Cuando un cliente entra mira si el barbero está libre (lo despierta) u ocupado. Si está ocupado se va a la sala de espera. Si la sala de espera está llena se marcha. **(3-8)**.

Inicialmente el barbero duerme mientras no tiene clientes. Cuando llega un cliente, despierta al barbero y se corta el pelo. Si llegan clientes en ese momento esperan en la sala de espera. Si llega otro cliente con la sala de espera llena se marcha. Cuando se termina el corte de pelo el barbero llama a uno de los que esperan. Si no hay duerme.

Problemas a solucionar

Sincronización entre barberos y cliente -> Un cliente llega al mismo tiempo que el barbero termina: mientras el cliente va a la sala de espera, el barbero la comprueba y la ve vacía: los dos esperan indefinidamente.

```
//Barbero
while(1) {
    mtx_lock(&mutex);
    if (!customers_waiting) {
        barber_state=SLEEPING;
        cnd_wait(&no_customers, &mutex);
    } else {
        cnd_signal(&waiting_room);
        waiting_customers--;
    }
    mtx_unlock(&mutex);
    cut_hair();
}

//Cliente
mtx_lock(&mutex);
if(waiting_customers==MAX_CUSTOMERS) {
    mtx_unlock(&mutex);
} else {
    if(barber_state==SLEEPING) {
        cnd_signal(&no_customers);
        barber_state=WORKING;
    } else {
        waiting_customers++;
        cnd_wait(&waiting_room, &mutex);
    }
    mtx_unlock(&mutex);
    get_hair_cut();
}
```

Solución: Observaciones

El barbero pone su estado a SLEEPING, pero es el cliente que lo despierta quien lo pone a WORKING. ¿Por que? => porque el cliente va a soltar el mutex, y podría venir otro cliente y creer que el barbero está libre. Hay que cambiar el estado antes de soltar el mutex.

¿Por qué se incrementan los clientes en espera en el cliente, pero se decrementan en el barbero? -> por el mismo motivo. Si llega un cliente después de que el barbero despierte a un cliente y no ha decrementado el contador, podría irse por pensar que la sala de espera está llena.

Múltiples Barberos

Dos estrategias:

-Añadir un contador de barberos libres en vez del estado. Los barberos incrementan el contador al irse a dormir, y los clientes lo decrementan a despertar un barbero.

-Guardar el estado de cada barbero. Los barberos ponen su estado a SLEEPING, y los clientes buscan un barbero que esté SLEEPING y lo ponen a WORKING. Hace falta una condición para dormir a cada barbero.

```
//Barbero
while(1) {
    mtx_lock(&mutex);
    if (!waiting_customers) {
        free_barbers++;
        cnd_wait(&no_customers, &mutex);
    } else {
        cnd_signal(&waiting_room);
        waiting_customers--;
    }
    mtx_unlock(&mutex);
    cut_hair();
}

//Cliente
mtx_lock(&mutex);
if(waiting_customers==MAX_CUSTOMERS) {
    mtx_unlock(&mutex);
} else {
    if(free_barbers>0) {
        cnd_signal(&no_customers);
        free_barbers--;
    } else {
        waiting_customers++;
        cnd_wait(&waiting_room, &mutex);
    }
    mtx_unlock(&mutex);
    get_hair_cut();
}
```

Con Semáforos

Vamos a usar dos semáforos:

-Uno con la cuenta de clientes esperando.

-Otro con el nº de barberos libres.

```
//Semáforos
sem_t customers;
sem_t barber;
sem_t free_seats;
int free_seats=N;
```

```
sem_init(&customers, 1, 0);
sem_init(&barber, 1, 0);
sem_init(&free_seats, 1, 1);

//Barbero
while(1) {
    sem_wait(&customers); // Wait for customers
    sem_wait(&free_seats); // lock counter
    free_seats++;
    sem_post(&barber); // Wake up a customer
    sem_post(&free_seats); // unlock counter
    cut_hair();
}

//Cliente
sem_wait(&free_seats); // lock counter
if(free_seats>0) {
    free_seats--;
    sem_post(&customer); // Increase available customers
    sem_post(&free_seats); // unlock counter
    sem_wait(&barber); // Wait for a barber
    get_hair_cut();
} else {
    sem_post(&free_seats);
}
```