

Barbero Durmiente

Concurrencia y Paralelismo

Juan Quintela

`quintela@udc.es`

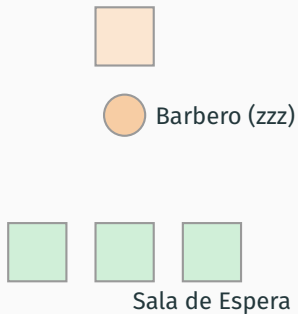
Javier París

`javier.paris@udc.es`

Descripción

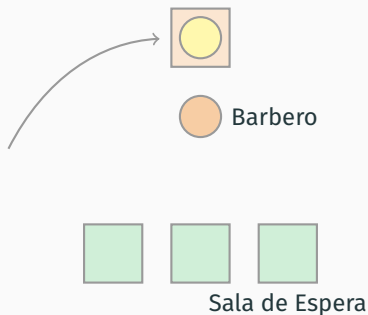
- 1 Barbero que atiende clientes.
- Al terminar de atender un cliente, el barbero comprueba la sala de espera, y si hay algún cliente esperando lo atiende.
- Si no hay clientes en la sala de espera, se pone a dormir.
- Cuando un cliente entra mira si el barbero está libre (lo despierta) u ocupado.
-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

Descripción



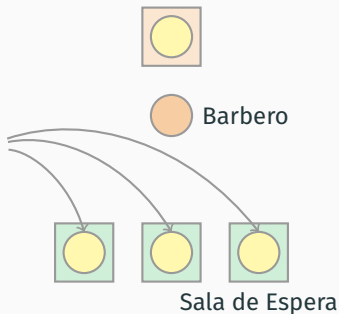
Inicialmente el barbero duerme mientras no tiene clientes

Descripción



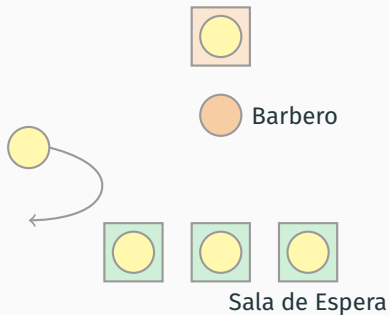
Cuando llega un cliente, despierta al barbero y se corta el pelo

Descripción



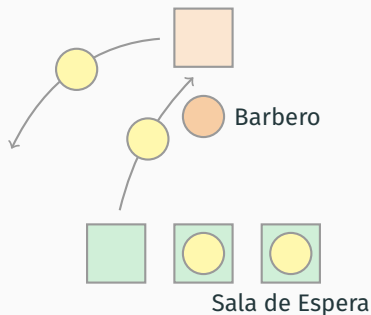
Si llegan clientes en ese momento esperan en la sala de espera

Descripción



Si llega otro cliente con la sala de espera llena se marcha.

Descripción



Cuando se termina el corte de pelo el barbero llama a uno de los que esperan. Si no hay duerme.

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.

Solución: Barbero

Barbero

```
while(1) {  
    pthread_mutex_lock(&mutex);  
    if (!customers_waiting) {  
        barber_state = SLEEPING;  
        pthread_cond_wait(&no_customers, &mutex);  
    }  
    else {  
        pthread_cond_signal(&waiting_room);  
        waiting_customers--;  
    }  
    pthread_mutex_unlock(&mutex);  
    cut_hair();  
}
```

Cliente

```
pthread_mutex_lock(&mutex);
if(waiting_customers == MAX_CUSTOMERS)
    pthread_mutex_unlock(&mutex);
else {
    if(barber_state==SLEEPING) {
        pthread_cond_signal(&no_customers);
        barber_state=WORKING;
    } else {
        waiting_customers++;
        pthread_cond_wait(&waiting_room, &mutex);
    }
    pthread_mutex_unlock(&mutex);
    get_hair_cut();
}
```

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 que 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.

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.

Múltiples Barberos: Barbero

Barbero

```
while(1) {  
    pthread_mutex_lock(&mutex);  
    if (!waiting_customers) {  
        barberos_libres++;  
        pthread_cond_wait(&no_customers, &mutex);  
    } else {  
        pthread_cond_signal(&waiting_room);  
        waiting_customers--;  
    }  
    pthread_mutex_unlock(&mutex);  
    cut_hair();  
}
```

Múltiples Barberos: Cliente

Cliente

```
pthread_mutex_lock(&mutex);
if(waiting_customers == MAX_CUSTOMERS)
    pthread_mutex_unlock(&mutex);
else {
    if(barberos_libres>0) {
        pthread_cond_signal(&no_customers);
        barberos_libres--;
    } else {
        waiting_customers++;
        pthread_cond_wait(&waiting_room, &mutex);
    }
    pthread_mutex_unlock(&mutex);
    get_hair_cut();
}
```

Con Semáforos

Vamos a usar dos semáforos:

- Uno con la cuenta de clientes esperando.
- Otro con el número 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); // Esperar por clientes  
    sem_wait(&free_seats); // bloquear contador  
    free_seats++;  
    sem_post(&barber); // Despertar un cliente  
    sem_post(&free_seats); // desbloquear contador  
    cut_hair();  
}
```

Cliente

```
sem_wait(&free_seats); // bloquear contador
if(free_seats>0) {
    free_seats--; // marcar silla ocupada
    sem_post(&customer); // Incrementar clientes
    sem_post(&free_seats); // soltar contador
    sem_wait(&barber); // Esperar por un barbero libre
    get_hair_cut();
} else sem_post(&free_seats);
```