2023

Práctica 7

ÁLVARO MARTÍNEZ GARCÍA
48150407-E

PROGRAMACIÓN CONCURRENTE

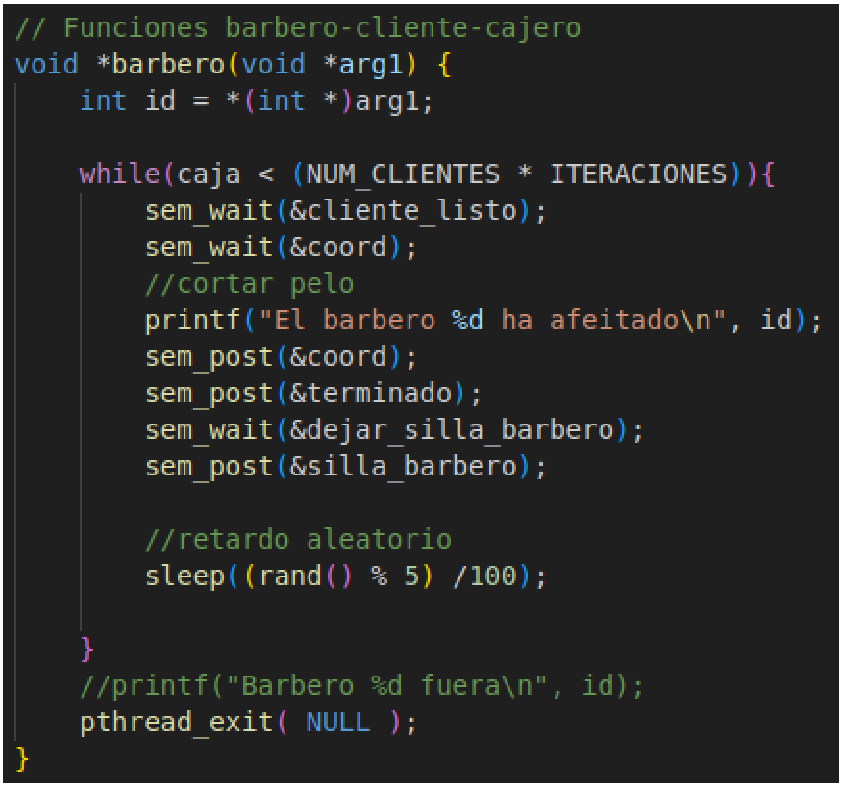
Contenido

Explicación

El código de esta práctica se encuentra en el archivo *practica7.c*. Este código es una adaptación, en C, del problema clásico de semáforos de la barbería. En este problema nos encontramos una barbería con 3 barberos, 3 sillas de barbero, un sofá de 4 plazas, una caja y un aforo máximo de clientes de 20 (13 estarían de pie esperando). Como los detallas de cómo funciona este problema ya se mencionan en el enunciado, en esta memoria se pasará a explicar directamente el código.

En primer lugar, tendremos que crear hilos que se encarguen de cumplir las funciones requeridas por el problema. En nuestro caso crearemos 25 hilos para clientes (para que haya clientes esperando "fuera de la tienda" para poder entrar cuando salga otro), 3 hilos para los 3 barberos y un hilo encargado de la caja. También tendremos los semáforos especificados en el enunciado e inicializados a los valores correspondientes. Por último, se utilizará una variable global *caja* para poder tener una variable que nos permita comprobar el correcto funcionamiento del programa, costando cada corte de pelo 1€. Teniendo en cuenta que tenemos 25 clientes y cada uno de ellos hará 2 iteraciones de afeitarse, el valor de la caja debería ser de 50€ al terminar la ejecución. De esta forma, podremos controlar la ejecución de los hilos de la caja y de los clientes, que terminarán cuando se cumplan esas condiciones.

A continuación, se explicará el funcionamiento de los 3 tipos de hilo de este programa. El primero es el encargado de los barberos, que empezarán esperando a que un cliente les avise que estén listos con el semáforo *cliente_listo*, que ocurrirá cuando haya tomado una de las sillas de barbero. Después, dependiendo del estado del semáforo *coord*, podría hacerse una transacción en caja o que el barbero continue con su ejecución y corte el pelo al cliente. Cuando termine con el cliente, esperará a que el cliente abandone la silla de barbero y notificará a los otros clientes que hay una silla de barbero libre. Al final de este bucle, se añade un pequeño retardo con el que atrasaremos la comprobación de la variable *caja* del bucle while del barbero y puedan terminar los barberos su ejecución cuando terminen los clientes.



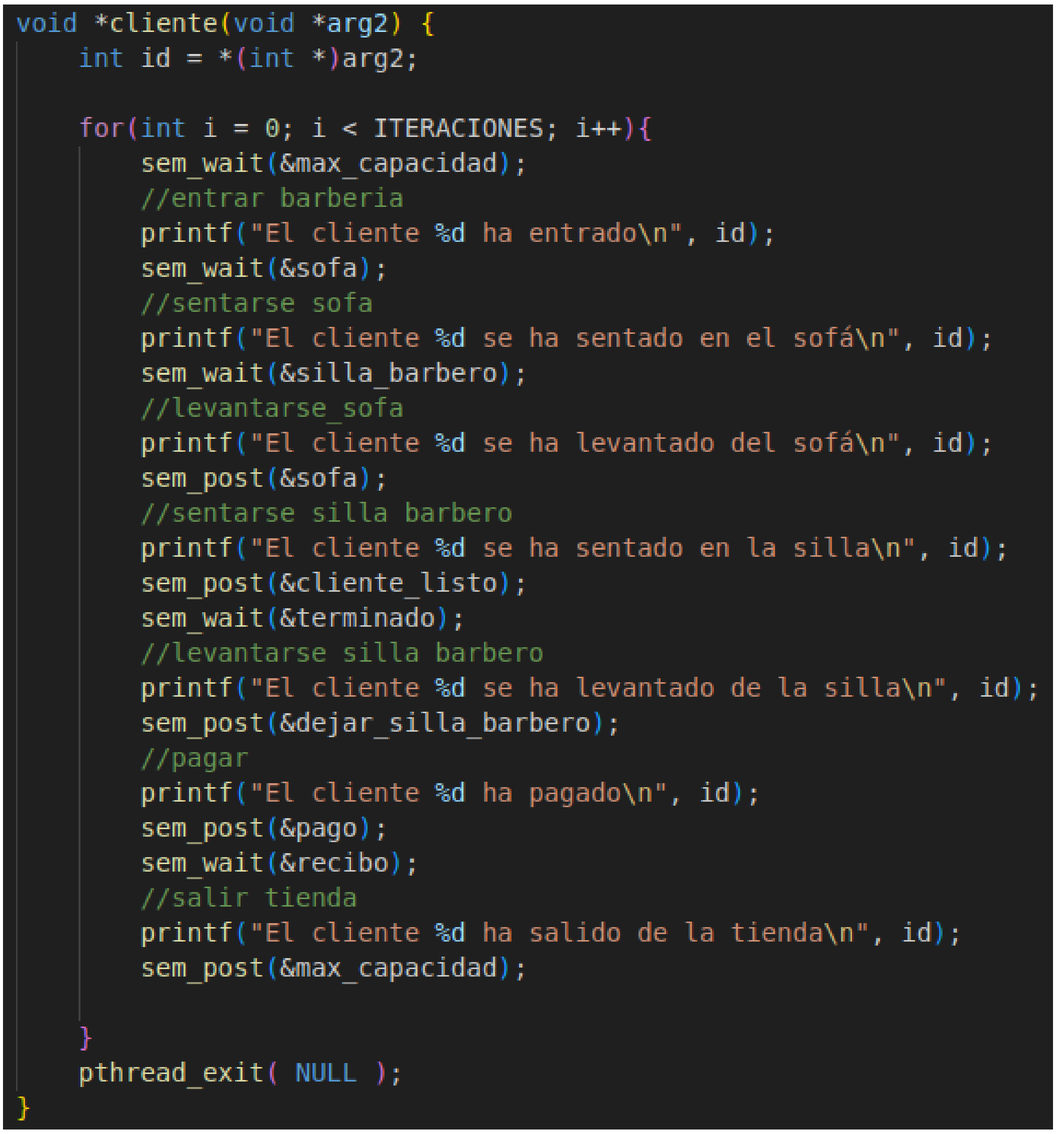
```
// Funciones barbero-cliente-cajero
void *barbero(void *arg1) {
    int id = *(int *)arg1;

    while(caja < (NUM_CLIENTES * ITERACIONES)){
        sem_wait(&cliente_listo);
        sem_wait(&coord);
        //cortar pelo
        printf("El barbero %d ha afeitado\n", id);
        sem_post(&coord);
        sem_post(&terminado);
        sem_wait(&dejar_silla_barbero);
        sem_post(&silla_barbero);

        //retardo aleatorio
        sleep((rand() % 5) /100);

    }
    //printf("Barbero %d fuera\n", id);
    pthread_exit( NULL );
}
```

El segundo tipo de hilos se encarga de los clientes. Los clientes primero entrarán a la tienda siempre que la capacidad máxima se lo permita. Una vez dentro esperarán a poder sentarse en el sofá. Cuando se hayan sentado en el sofá, significa que es uno de los clientes que lleva más tiempo esperando, por lo que ahora puede esperar a que una silla de barbero esté libre. Cuando esta silla esté libre, el cliente se levantará del sofá y notificará este espacio libre para que lo tome otro hilo esperando en el semáforo de sofá. Cuando el cliente se siente, notificará al barbero con el semáforo *cliente_listo* y esperará a que el barbero termine con él. Una vez haya terminado, le indicará al barbero, con el semáforo *dejar_silla_barbero* que el cliente ha dejado la silla, pagará en la caja, esperará al recibo y, por último, saldrá de la tienda y avisará a otros clientes la posibilidad de entrar.

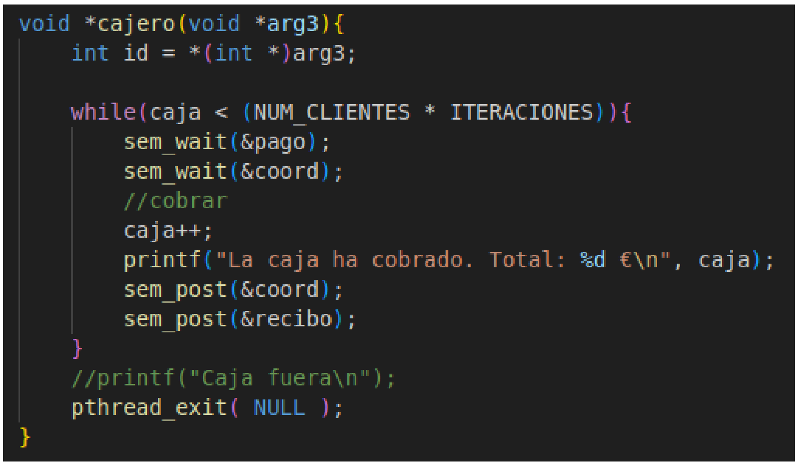


```c
void *cliente(void *arg2) {
    int id = *(int *)arg2;

    for(int i = 0; i < ITERACIONES; i++){
        sem_wait(&max_capacidad);
        //entrar barberia
        printf("El cliente %d ha entrado\n", id);
        sem_wait(&sofa);
        //sentarse sofa
        printf("El cliente %d se ha sentado en el sofá\n", id);
        sem_wait(&silla_barbero);
        //levantarse_sofa
        printf("El cliente %d se ha levantado del sofá\n", id);
        sem_post(&sofa);
        //sentarse silla barbero
        printf("El cliente %d se ha sentado en la silla\n", id);
        sem_post(&cliente_listo);
        sem_wait(&terminado);
        //levantarse silla barbero
        printf("El cliente %d se ha levantado de la silla\n", id);
        sem_post(&dejar_silla_barbero);
        //pagar
        printf("El cliente %d ha pagado\n", id);
        sem_post(&pago);
        sem_wait(&recibo);
        //salir tienda
        printf("El cliente %d ha salido de la tienda\n", id);
        sem_post(&max_capacidad);

    }
    pthread_exit( NULL );
}
```

Por último, el hilo de la caja es más sencillo que los anteriores. Primero tiene dos semáforos, *pago* y *recibo*, que se encargan de esperar a que un cliente pague y que un barbero venga a cobrar la caja, respectivamente. Después, aumentará la variable *caja* en 1 y avisará al barbero y al cliente con los semáforos *coord* y *recibo* que la transacción ha terminado. Este bucle se repetirá hasta que haya en la caja el número de clientes multiplicado por el número de iteraciones de cortes de pelo, que en este código se ha indicado como 2.



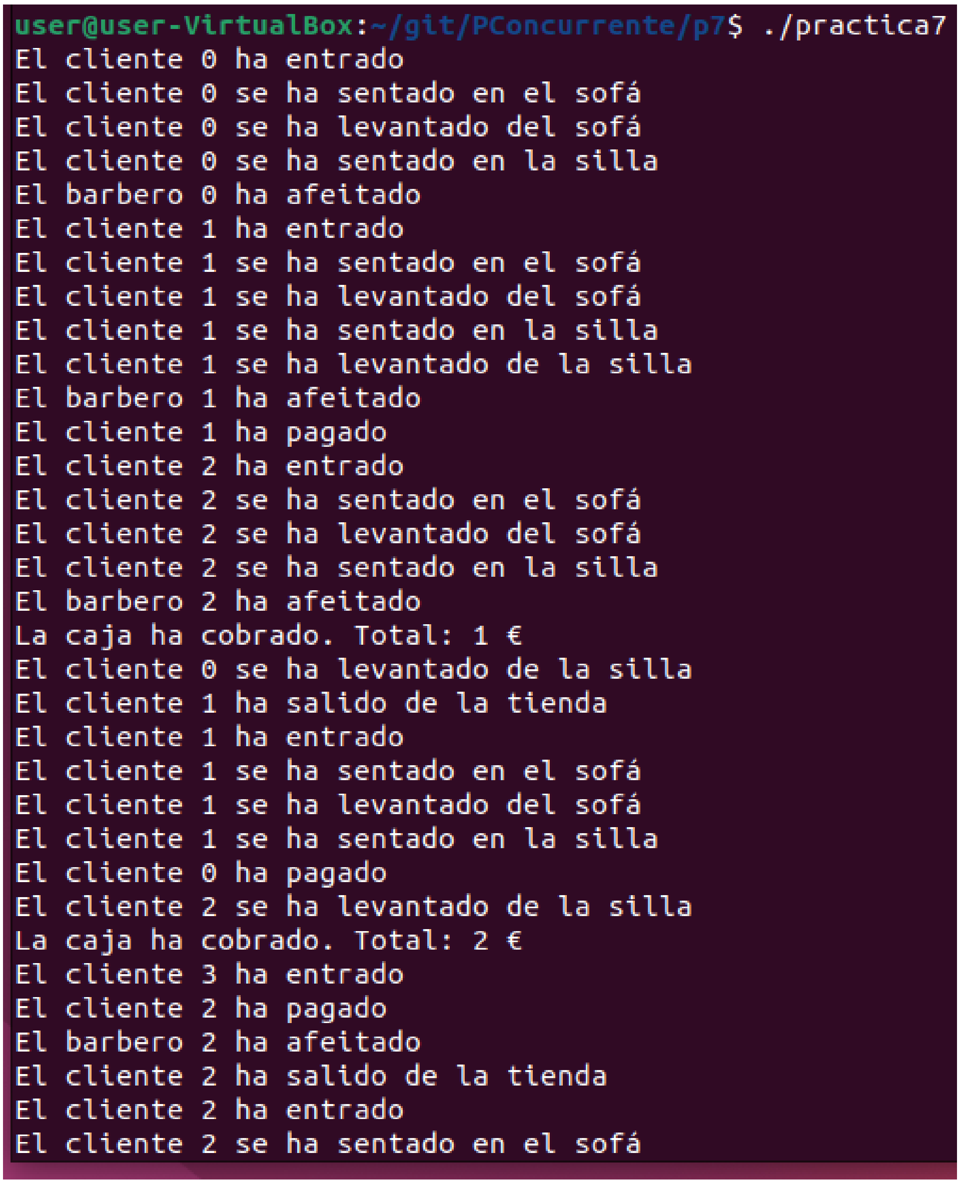
```c
void *cajero(void *arg3){
    int id = *(int *)arg3;

    while(caja < (NUM_CLIENTES * ITERACIONES)){
        sem_wait(&pago);
        sem_wait(&coord);
        //cobrar
        caja++;
        printf("La caja ha cobrado. Total: %d €\n", caja);
        sem_post(&coord);
        sem_post(&recibo);
    }
    //printf("Caja fuera\n");
    pthread_exit( NULL );
}
```

Antes de terminar la ejecución, hay que hacer los *join* de los hilos y notificar que, al no haber más clientes, la barbería se cerrará y terminará la ejecución de nuestro programa.

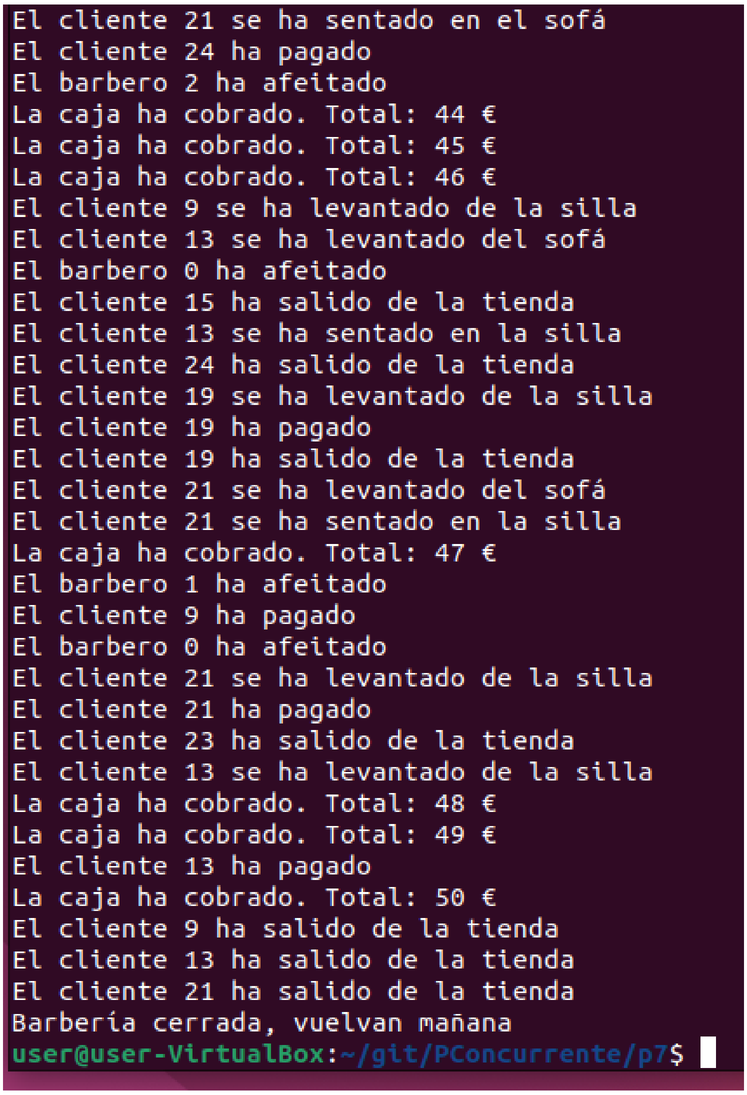
Salida

Para comprobar que el programa funciona de forma correcta, para cada uno de los métodos representados en el pseudocódigo del enunciado, se mostrará una salida en pantalla que nos avise de ello. Además, la salida del método *cobrar()* en el hilo de caja, mostrará la cantidad de la caja, pudiendo comprobar nosotros que el programa funciona correctamente y llega a 50€ como se tiene previsto para este programa. Si ejecutamos el programa, podemos observar en la salida (no toda al ser bastantes líneas de salida como para mostrarlas aquí) lo siguiente:



```
user@user-VirtualBox:~/git/PConcurrente/p7$ ./practica7
El cliente 0 ha entrado
El cliente 0 se ha sentado en el sofá
El cliente 0 se ha levantado del sofá
El cliente 0 se ha sentado en la silla
El barbero 0 ha afeitado
El cliente 1 ha entrado
El cliente 1 se ha sentado en el sofá
El cliente 1 se ha levantado del sofá
El cliente 1 se ha sentado en la silla
El cliente 1 se ha levantado de la silla
El barbero 1 ha afeitado
El cliente 1 ha pagado
El cliente 2 ha entrado
El cliente 2 se ha sentado en el sofá
El cliente 2 se ha levantado del sofá
El cliente 2 se ha sentado en la silla
El barbero 2 ha afeitado
La caja ha cobrado. Total: 1 €
El cliente 0 se ha levantado de la silla
El cliente 1 ha salido de la tienda
El cliente 1 ha entrado
El cliente 1 se ha sentado en el sofá
El cliente 1 se ha levantado del sofá
El cliente 1 se ha sentado en la silla
El cliente 0 ha pagado
El cliente 2 se ha levantado de la silla
La caja ha cobrado. Total: 2 €
El cliente 3 ha entrado
El cliente 2 ha pagado
El barbero 2 ha afeitado
El cliente 2 ha salido de la tienda
El cliente 2 ha entrado
El cliente 2 se ha sentado en el sofá
```

En esta primera captura del principio de la ejecución del programa, podemos observar como los primeros clientes entran a la tienda y siguen el procedimiento de entrar, sentarse en el sofá, sentarse en la silla, terminar e irse. Además, aunque es unas líneas más debajo de cuando se muestra que ha pagado, por ejemplo, el cliente 1, se puede ver cómo el total de la caja aumenta en 1. Los otros clientes siguen el mismo procedimiento de la misma manera en todas las ejecuciones del programa.



```
El cliente 21 se ha sentado en el sofá
El cliente 24 ha pagado
El barbero 2 ha afeitado
La caja ha cobrado. Total: 44 €
La caja ha cobrado. Total: 45 €
La caja ha cobrado. Total: 46 €
El cliente 9 se ha levantado de la silla
El cliente 13 se ha levantado del sofá
El barbero 0 ha afeitado
El cliente 15 ha salido de la tienda
El cliente 13 se ha sentado en la silla
El cliente 24 ha salido de la tienda
El cliente 19 se ha levantado de la silla
El cliente 19 ha pagado
El cliente 19 ha salido de la tienda
El cliente 21 se ha levantado del sofá
El cliente 21 se ha sentado en la silla
La caja ha cobrado. Total: 47 €
El barbero 1 ha afeitado
El cliente 9 ha pagado
El barbero 0 ha afeitado
El cliente 21 se ha levantado de la silla
El cliente 21 ha pagado
El cliente 23 ha salido de la tienda
El cliente 13 se ha levantado de la silla
La caja ha cobrado. Total: 48 €
La caja ha cobrado. Total: 49 €
El cliente 13 ha pagado
La caja ha cobrado. Total: 50 €
El cliente 9 ha salido de la tienda
El cliente 13 ha salido de la tienda
El cliente 21 ha salido de la tienda
Barbería cerrada, vuelvan mañana
user@user-VirtualBox:~/git/PConcurrente/p7$
```

En esta captura podemos ver las últimas líneas de la ejecución anterior. En ella podemos observar a los últimos clientes saliendo de la tienda, como el total de la caja llega a 50€ como debería y como, cuando sale el último cliente, la barbería cierra.