Ángel Barrilao Bensrhir - Grupo 2º A

Práctica 1

Problema 1: Productor - Consumidor

Explicación sobre uso de semáforos

Para resolver el problema del productor-consumidor se han usado tres semáforos.

Los semáforos son : uno para producir , otro para consumir y finalmente otro de exclusión mutua para el acceso al buffer apodado mutex.

El orden en el que se deben poner los semáforos seria el siguiente :

función consumidor :

```
sem_wait(&consumir);
sem_wait(&mutex);
        dato=buffer[ocupacion-1];
        ocupacion--;
sem_post(&mutex);
sem_post(&producir);
```

función producir:

```
sem_wait(&producir);
sem_wait(&mutex);
        buffer[ocupacion]=dato;
        ocupacion++;
sem_post(&mutex);
sem_post(&consumir);
```

Fíjese que según entremos en la función, enviaremos la señal con sem_wait al semáforo correspondiente para que empiece a trabajar su parte critica sin interrupciones.

Un punto muy importante es que controlemos el acceso a la sección critica para que sea individual. Ya que se producirían incoherencias al acceder varios hilos al mismo tiempo del que se escribe o lee en el buffer.

Finalmente liberamos los semáforos con sem_post y cuando finaliza la ejecuccion los destruimos con sem_destroy

Problema 2: Fumadores

Lo primero que se ha realizado es declarar las variables , hay una pequeña variación con respecto al productor – consumidor y es en la declaración de los semáforos.

En este caso para cada fumador va a tener su propio semáforo y lo declaramos como un array de semáforos, ya que cada fumador es independiente y cada uno se irá ejecutando según lo que valla produciendo el estanquero.

```
//Semaforos
sem_t fumadores[3];
sem_t estanquero;
sem_t mutex;
```

En la **función_estanquero** lo primero que vamos a hacer es definir un bucle infinito, a continuación generamos un numero aleatorio entre 3 - 0

Seguidamente bloquemos el turno del estanquero , bloquemos el mutex y mostramos el mensaje por pantalla . El mensaje va en la sección critica para asegurarnos de que se va a ejecutar sin ninguna interrupción porque de lo contrario se cortaría o perdería en el transcurso de la ejecución.

Finalmente liberamos los semáforos para el próximo mutex y para un fumador en especifico , según el ingrediente generado.

```
void * funcion_estanquero(void *){
    while (true){
        /*incluye el cero pero excluye el maximo(3)
        es decir llega hasta 2.*/
        unsigned int ingrediente_aleatorio=(rand() % 3 + 0);//Producimos ingr. aleatorio

        sem_wait(&estanquero);//Para la prox. estanquero esta parado
        sem_wait(&mutex);//Bloquemos secc. critica
                cout<<"Estanquero genera: "<<ingrediente_aleatorio<<endl;
        sem_post(&mutex);//Desbloqueamos secc. critica
        sem_post(&fumadores[ingrediente_aleatorio]);/*Avisamos a fumador
        segun el ingrediente aleatorio generado  */

    }
}
```

Por último la **función_fumador** recibe el id del fumador y lo que hacemos es bloquear el susodicho fumador y desbloqueamos al estanquero para que empiece a producir y otras hebras de fumadores sigan consumiendo.

Después llamamos a la función fumar ,esta mete un retardo y mientras este 'fumando' , el estanquero ya habrá producido mas ingredientes y otros fumadores estarán fumando a la par.

```
void * funcion_fumador(void* num_fum_void){
    unsigned long ih= (unsigned long) num_fum_void;

    while (true){
        sem_wait(&fumadores[ih]);//Bloquemos el fumador especificado
            //*************VACIO*****************
        sem_post(&estanquero);//avisamos al estanquero para que empieze a producir
                fumar(ih);//Empieza a fumar
    }


}
```