Fumadores

Los semaforos utilizados son :

puede_producir : se utiliza para controlar la generacion de los ingredientes.

Su valor inicial es 1

El sem_wait se hace antes de generar el ingrediente, y sem_post despues de empezar la accion de fumar.

fumadores[n_fumadores] : se utiliza para controlar que fumador va a realizar la accion de fumar.

Su valor inicial es 0.

El sem_wait se hace antes de fumar, y sem_post despues de generado el ingrediente.

mutex : se utiliza para sincronizar las salidas.

Su valor inicial es 1.

El sem_wait se utiliza antes de los cout, y el sem_post despues de los cout.

Codigo:

```
#include <iostream>
#include <cassert>
#include <pthread.h>
#include <semaphore.h>
#include <time.h>           // incluye "time(....)"
#include <unistd.h>       // incluye "usleep(...)"
#include <stdlib.h>       // incluye "rand(...)" y "srand"

using namespace std;
// ----------------------------------------------------------
------------------
int material, veces=20;
const int n_fumadores=3;
sem_t puede_producir;
sem_t fumadores[n_fumadores];
sem_t mutex;


// funci 贸 n que simula la acci 贸 n de fumar  como un retardo
aleatorio de la hebra
```

```
void    fumar()
{
    //    inicializa la semilla aleatoria    (solo la primera vez)
    static bool primera_vez = true ;
    if ( primera_vez )
    {     primera_vez = false ;
          srand( time(NULL) );
    }

    // calcular un numero aleatorio de milisegundos (entre 1/10 y 2
segundos)
    const unsigned miliseg = 100U + (rand() % 1900U) ;

    // retraso bloqueado durante 'miliseg' milisegundos
    usleep( 1000U*miliseg );
}
// ----------------------------------------------------------
------------------

// falta: resto de funciones
// ..............
```

```
void * estanquero(void *){

    for(int i=0;i<veces;i++){

        sem_wait(&puede_producir);
        material=(rand() % 3U);    //1+rand()%(4-1)

            sem_wait(&mutex);
            cout<<"Se ha generado el material " <<material+1 <<endl;
            sem_post(&fumadores[material]);
            sem_post(&mutex);
    }
    return NULL;
}


void * fumador(void *){
    for(int i=0;i<veces;i++){

        sem_wait(&fumadores[material]);
        int fumador=material+1;
```

```
        sem_wait(&mutex);
        cout<<"El fumador "<< fumador << " esta fumando con material " <<material+1 <<endl;

        fumar();
        sem_post(&puede_producir);
        cout<<"El fumador "<< fumador << " ha terminado de fumar"<<endl;
        sem_post(&mutex);
    }
    return NULL;
}

// ------------------------------------------------------------------

int main()
{
   // falta: creaci 贸 n hebras ....
   sem_init(&mutex, 0, 1);
   sem_init(&puede_producir,0,1);
```

```
  for(unsigned i=0;i<n_fumadores;i++){
    sem_init(&(fumadores[i]),0,0);
  }


  pthread_t h_estanquero,h_fumadores[n_fumadores];

  pthread_create(&h_estanquero,NULL,estanquero,NULL);
  for(unsigned i=0;i<n_fumadores;i++){
    pthread_create(&(h_fumadores[i]),NULL,fumador,NULL);
  }

  pthread_join(h_estanquero,NULL);
for(unsigned i=0;i<n_fumadores;i++){
    pthread_join(h_fumadores[i],NULL);
}

  sem_destroy(&mutex);
  sem_destroy(&puede_producir);
    for(unsigned i=0;i<n_fumadores;i++){
    sem_destroy(&(fumadores[i]));
  }
```

```
    pthread_exit(NULL);


    return 0 ;
}
```