**EJERCICIO 1**

Indique que realiza el siguiente código:

|  |  |
| --- | --- |
| #include <stdio.h>  #include <stdlib.h>  #include <pthread.h>  #define N 10  #define TAMANIO 1024  void \*trabajador(void \*arg);  int vector[TAMANIO];  struct b\_s {  int n;  pthread\_mutex\_t m;  pthread\_cond\_t ll;  } b;  int main(void) {  pthread\_t hilo[N];  int i;    b.n = 0;  pthread\_mutex\_init(&b.m, NULL);  pthread\_cond\_init(&b.ll, NULL);  par=0; impar=1;    for(i=0; i<N; i++)  pthread\_create(&hilo[i],  NULL, trabajador,  (void \*)&i);    for(i=0; i<N; i++)  pthread\_join(hilo[i], NULL);  pthread\_cond\_destroy(&b.ll);  pthread\_mutex\_destroy(&b.m);  return 0;  } | void \*trabajador(void \*arg) {  int inicio=0, fin=0, i;  id = \*(int \*)arg;  inicio =(id)\*(TAMANIO/N);  fin = (id+1)\*(TAMANIO/N);  for(i=inicio; i<fin; i++) {  vector[i] = id;  }  pthread\_mutex\_lock(&b.m);  b.n++;  if (N<=b.n) { pthread\_cond\_broadcast(&b.ll);  } else {  pthread\_cond\_wait(&b.ll, &b.m);  }  pthread\_mutex\_unlock(&b.m);    return 0;  } |

¿Se pueden producir condiciones de carrera?

Incluya los semáforos necesarios para que dicha condición de carrera no se produzca.

Nota: para comprobarlo, podéis incluir un printf en cada trabajador, que escriba el número de hilo que está trabajando y el valor del identificador , y de las variables locales inicio y fin.

**EJERCICIO 2**

Implementar un programa que resuelva el problema del productor-consumidor con MUTEX. El programa describe dos thread, productor y consumidor, que comparten un [buffer](https://es.wikipedia.org/wiki/Buffer) de tamaño finito. La tarea del productor es generar un número entero, almacenarlo y comenzar nuevamente; mientras que el consumidor toma (simultáneamente) números uno a uno. El problema consiste en que el productor no añada más números que la capacidad del buffer y que el consumidor no intente tomar un número si el buffer está vacío.

**EJERCICIO 3**

Implementar un programa que resuelva el problema del productor-consumidor con Semáforos POSIX. El programa describe dos thread, productor y consumidor, que comparten un [buffer](https://es.wikipedia.org/wiki/Buffer) de tamaño finito. La tarea del productor es generar un número entero, almacenarlo y comenzar nuevamente; mientras que el consumidor toma (simultáneamente) números uno a uno. El problema consiste en que el productor no añada más números que la capacidad del buffer y que el consumidor no intente tomar un número si el buffer está vacío.