- SISTEMAS OPERATIVOS -
1º GRADO INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE CÓRDOBA

Práctica II – Introducción a hilos & Productor-Consumidor:

Indispensables:

- El tipo estándar de semáforos contadores POSIX es sem_t. Para usar los semáforos en diferentes hilos los declararemos como variables globales. Para usarlos en diferentes procesos los alojaremos en memoria compartida.

```
#include <semaphore.h>
sem_t mutex;
```

- Para inicializar y usar un semáforo llamado "mutex" para proteger una sección crítica haríamos:

```
sem_t mutex;
sem_init (&mutex, 0, 1);
sem_wait (&mutex)
// Sección crítica
sem_post (&mutex);
```

- El segundo argumento de sem_init () debe valer 0 si el semáforo va a ser usado por diferentes hilos de un mismo proceso, y debe valer 1 si se va a usar por diferentes procesos.

Ejercicios propuestos:

1. Compilar y ejecutar el programa hilos.c.
   1.1. Leer y reflexionar sobre las diferencias entre procesos e hilos.
   1.2. Comprobar que el resultado de la suma no es correcto e identificar la sección crítica del programa.
2. Modificar el programa para que ahora se ejecuten 4 hilos y aplicar el algoritmo de Lamport para resolver el problema de la sección crítica del programa.
3. Los hilos productores generan números aleatorios en el intervalo [0,1000] y los depositan en el búfer limitado de tamaño N. Los hilos consumidores leen los números depositados en el búfer y realizan las sumas sucesivas de los números leídos. Para comprobar la corrección de la solución programada los productores realizarán las sumas de los números que producen para comprobar si las sumas coinciden con las de los consumidores.
   - Solucionar el problema de programación concurrente según las siguientes formulaciones:

   3.1. Hay un único hilo productor y un único hilo consumidor.
   3.2. Hay P hilos productores y C hilos consumidores. Cada número generado por un productor es consumido por un único consumidor. Esta es la formulación estándar del problema del productor consumidor.
   3.3. Hay un hilo productor y C hilos consumidores. Cada número generado por el productor ha de ser consumido por TODOS los hilos consumidores.

Prototipos hilos (*#include <pthread.h>*, compilar con *-lpthread*):

```
--------------------------
int pthread_create(pthread_t * thread, pthread_attr_t *attr, void *
(*start_routine) (void *),void *arg)
int pthread_join(pthread_t th, void **thread_return)
void pthread_exit (void *retval)
```

Prototipos semáforos (*#include <semaphore.h>*):

```
--------------------------
int sem_init(sem_t *sem, int pshared, int value);
int sem_wait(sem_t *sem);
int sem_post(sem_t *sem);
```

Pseudocódigo algoritmo de Lamport:

```
// Variables globales compartidas
BOOLEAN Eligiendo[N];
int Numero[N];

// Inicialización de variables
Eligiendo[] = {FALSE, FALSE,..., FALSE};
Numero[] = {0, 0,..., 0};

// Proceso o hilo I-ésimo
void Proceso(int I)
{
    extern BOOLEAN Eligiendo[N];
    extern int Numero[N];
    int j;

    while (TRUE) {
      Eligiendo[I] = TRUE;
      Numero[I] = max(Numero[0], Numero[1],..., Numero[N - 1]) + 1;
      Eligiendo[I] = FALSE;
      for (j = 0; j < N; j++) {
          while (Eligiendo[j]);
          while ((Numero[j] != 0) && (Numero[j], j) < (Numero[I], I));
      }

      //--------------------
      // SECCIÓN CRÍTICA
      //--------------------

      Numero[I] = 0;

      // Sección residual
    }
}
```

Pseudocódigo productor-consumidor:

```
// Variables globales compartidas
item bufer[N];
semaforo empty, full, mutex;

// Inicializacion de variables
init(mutex, 1);
init(full, 0);
init(empty, N);


// Proceso o hilo productor
void Productor()
{
    extern semaforo mutex, full, empty;
    T dato;

    while (TRUE) {
      ProducirDato(dato);
      wait(empty);
      wait(mutex);
      EntrarDato(dato);
      signal(mutex);
      signal(full);
    }
}

// Proceso o hilo consumidor
void Consumidor()
{
    extern semaforo mutex, full, empty;
    T dato;

    while (TRUE) {
      wait(full);
      wait(mutex);
      SacarDato(dato);
      signal(mutex);
      signal(empty);
      ConsumirDato(dato);
    }
}
```