# Informe Del Ejercicio Opcional 1

## 1. Introducción.

En este ejercicio hemos hecho una implementación basada en el uso de hilos y semáforos, solucionando el problema del productor-consumidor haciendo que el hilo productor y el hilo consumidor operen sin problemas en una estructura de datos compartida usando semáforos (el uso de hilos permite compartir memoria más fácilmente, ya que todos los hilos de un proceso comparten el mismo espacio de direcciones).

## 2. Ejecución.

La implementación en C está formada de un programa: prodConsHilos.c (programa que gestiona el funcionamiento de los dos hilos). Generando el ejecutable con este comando:

```
gcc -o prodConsHilos prodConsHilos.c -lpthread
```

Ejecutando el programa de esta manera:

```
./prodConsHilos
```

# 3. Funcionamiento.

Este programa implementa dos hilos: un productor y un consumidor, los cuales comparten una pila de tamaño limitado (en este caso 8), a la que acceden de manera ordenada, utilizando semáforos, con los que tienen un acceso controlado, evitando las condiciones de carrera. Se utilizan los semáforos: MUTEX, para controlar el acceso a la pila, VACÍAS, que indica cuántas posiciones vacías hay en la pila, y LLENAS, que nos indica cuántas posiciones de la pila están ocupadas. Teniendo 60 iteraciones que realizan cada hilo respectivamente, teniendo un funcionamiento análogo al de los programas (el productor y el consumidor) del ejercicio anterior. En el main se inicializan los semáforos y se crean los hilos del productor y del consumidor de esta manera:

```
// Crear hilos de productor y consumidor
pthread_t t_prod, t_cons;
pthread_create(&t_prod, NULL, productor, NULL);
pthread_create(&t_cons, NULL, consumidor, NULL);
```

Una vez creados los hilos se espera a que terminen usando:

```
// Esperar a que los hilos terminen
pthread_join(t_prod, NULL);
pthread_join(t_cons, NULL);
```

Para luego eliminar los semáforos.

Hemos ejecutado el programa cambiando el valor de las iteraciones a 20 para que fuera más visible:

```
joel@joel-Victus:~/Escritorio/uni/Sistemas Operativos II/practica2/codigos$ ./prodConsHilos
Productor: Introdujo D en la posición 0
Productor: Introdujo R en la posición 1
Consumidor: Retiró R de la posición 1
Consumidor: Retiró D de la posición 0
Productor: Introdujo R en la posición 0
Productor: Introdujo U en la posición 1
Productor: Introdujo U en la posición 2
Productor: Introdujo Y en la posición 3
Productor: Introdujo J en la posición 4
Productor: Introdujo L en la posición 5
Consumidor: Retiró L de la posición 5
Consumidor: Retiró J de la posición 4
Consumidor: Retiró Y de la posición 3
Productor: Introdujo D en la posición 3
Productor: Introdujo M en la posición 4
Consumidor: Retiró M de la posición 4
Productor: Introdujo M en la posición 4
Consumidor: Retiró M de la posición 4
Productor: Introdujo L en la posición 4
Productor: Introdujo U en la posición 5
Consumidor: Retiró U de la posición 5
Productor: Introdujo E en la posición 5
Productor: Introdujo H en la posición 6
Consumidor: Retiró H de la posición 6
Productor: Introdujo D en la posición 6
Productor: Introdujo K en la posición 7
Consumidor: Retiró K de la posición 7
Consumidor: Retiró D de la posición 6
Consumidor: Retiró E de la posición 5
Productor: Introdujo U en la posición 5
Consumidor: Retiró U de la posición 5
Productor: Introdujo F en la posición 5
Consumidor: Retiró F de la posición 5
Productor: Introdujo I en la posición 5
Productor produjo: DRRUUYJLDMMLUEHDKUFI
Consumidor: Retiró I de la posición 5
Consumidor: Retiró L de la posición 4
Consumidor: Retiró D de la posición 3
Consumidor: Retiró U de la posición 2
Consumidor: Retiró U de la posición 1
Consumidor: Retiró R de la posición 0
Consumidor consumió: RDLJYMMUHKDEUFILDUUR
```

En esta ejecución se observa que los hilos productor y consumidor funcionan paralelamente de manera correcta, mediante su sincronización usando semáforos, asegurándonos que el productor no inserte elementos en la pila cuando esté llena, y que el consumidor no retire elementos cuando esté vacía.

## 4. Conclusión.

Este ejercicio nos muestra una solución eficiente al problema del productor-consumidor utilizando hilos, lo que permite compartir memoria de manera más eficiente en comparación con los ejercicios anteriores, ya que evita tener que compartir memoria entre los procesos. Y usando los semáforos nos aseguramos un acceso ordenado y que no haya datos incorrectos.