# Práctica: Sincronización de procesos

Miguel Ángel Conde González
Antonio Gómez García
Mario Enrique Casado García

November 10, 2019

## Objetivos

- Manejo de procesos y mecanismos de sincronización entre ellos.
- Afianzar los conocimientos acerca del lenguaje C.
- Afianzar los conocimientos acerca de los sistemas operativos.

## Evaluación

- Se comprobará el correcto funcionamiento del programa.
- Será obligatorio enviar el código a través de agora.unileon.es.
- La práctica se evaluará teniendo en cuenta el funcionamiento de la misma, la claridad del código y la calidad de la solución aportada.

## Enunciado de la práctica

En este ejercicio se realizará un repaso de toda la materia vista hasta ahora así como a algunos aspectos avanzados de C.

La presente práctica va a constar de dos partes:

1. Un programa que el funcionamiento de una cocina de un restaurante. Téngase en cuenta que en el ejercicio se trabajará con varios procesos. Para poder seguir la actividad del programa, es imprescindible que cada proceso muestre trazas de lo que hace en cada momento, identificando claramente qué proceso hace cada cosa.

2. Un script Shell que presenta un menú de cuatro opciones.

   - Si se selecciona la primera, opción el script muestra el código del programa mediante el uso del comando *cat* (`cat programa.c`).
   - Si se selecciona la segunda, se lanzará la compilación del archivo .c en que entregue el programa (`gcc programa.c -o programa`).

- Si se escoge la tercera, se ejecutará el programa, siempre que exista el ejecutable y tenga permisos de ejecución. Para proceder a dicha ejecución se pedirá el número de pinches de cocina que luego se pasara como argumentos al programa.
- En caso de que se escoja la cuarta, se saldrá del script.

**Cocina de un restaurante**

El programa va a simular el funcionamiento de una cocina de un restaurante. En esta cocina existirá un chef (proceso principal), un sommelier, un jefe de sala y un mozo de los recados. Además de tantos pinches de cocina como se especifiquen en la entrada al programa. Todos estos procesos se consideran hijos del proceso principal (chef) excepto el mozo de los recados que lo será del somelier.

El chef (proceso principal) al lanzarse va a crear al somelier, al jefe de sala y a los pinches de cocina. Todos ellos deberán quedar esperando órdenes del chef. El sommelier creará como hijo el proceso mozo de los recados.

Cuando el chef va a empezar a prepar los platos del día de hoy debe asegurarse que dispone de todos los ingredientes y el vino, para ello duerme 3 segundos y genera un aleatorio, si es un 0, le faltan ingredientes y le manda la señal SIGUSR1 al somelier, sino solamente le falta el vino y le manda SIGUSR2. El somelier, según la señal recibida, enviará un aviso al mozo de los recados mediante la señal SIGPIPE para que vaya a buscar lo que falte, el mozo cuando haya finalizado devolverá al somelier valor en función de si encontró o no el vino o los ingredientes (1 si los encontró y 0 si no lo hizo). El somelier informará al chef de si falta vino (devolviendo 1), ingredientes (devolviendo 2) o ninguna de las dos cosas (devolviendo 3). Si falta vino el chef acabaría el programa siempre antes matando todos sus hijos, si faltan ingredientes escribiría un mensaje con esa información. Tanto en este último caso, como si no falta nada se avisaría a los pinches de cocina de que empiecen a preparar los platos y se quedará esperando por el resultado. Los pinches de cocina cuando reciban la señal de comenzar a preparar platos dormirán un número aleatorio de segundos entre 2 y 5 y generarán un valor con un 0 o un 1 dependiendo de si el plato se ha cocinado bien o no. Ese número se le devolverá al chef que, una vez finalicen todos los pinches. Éste contará cuantos platos se han completado, si el número de platos es 0 cerrará el restaurante sino imprimirá un mensaje al respecto del número de platos y avisará al jefe de sala (SIGUSR1) para que monte las mesas)
El jefe de sala cuando reciba la señal dormirá 3 segundos y escribirá que ha finalizado con el mensaje de las mesas. Una vez este proceso termine el chef escribirá "PUEDE ABRIRSE EL RESTAURANTE" y acabará el programa.

# Algunas funciones C

Para calcular números aleatorios en un intervalo puede utilizarse la siguiente función

```
int calculaAleatorios(int min, int max) {
    return rand() % (max-min+1) + min;
}
```

Sin embargo el uso de esa función requiere incluir la biblioteca de C stdlib.h y la iniciación de una semilla de números aleatorios con un número único. Para ello dentro de la función main del programa deberá utilizarse la siguiente sentencia:

```
srand (time(NULL));
```

Otra de las acciones que se debe hacer es que los procesos duerman (suspendan su ejecución) un número de segundos, para ello debe usarse la función sleep de C. Por ejemplo la siguiente llamada a sleep supondría que el proceso durmiera 10 segundos

```
sleep(10);
```

## Otros aspectos a tener en cuenta

- **ES IMPRESCINDIBLE QUE LA PRÁCTICA FUNCIONE CORRECTAMENTE**
  Si esto no se tiene en cuenta se obtendrá un 0 en esta práctica.
- No deben usarse variables globales.
- Los nombres de las variables deben ser inteligibles.
- El código debe estar indentado y deben usarse comentarios.
- Para esta práctica no deben utilizarse threads, solamente procesos y señales