Práctica: Sincronización de procesos

Miguel Ángel Conde González
Antonio Gómez García
Mario Enrique Casado García

November 4, 2020

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 un centro de experimentación de vacunas de COVID19 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 sujetos experimentales que luego se pasara como argumentos al programa.
- En caso de que se escoja la cuarta, se saldrá del script.

Consultorio COVID

El programa va a simular el funcionamiento de un consultorio para pruebas experimentales de COVID. En él el epidemiólogo va a proceder a probar vacunas y para ello requiere de un médico y un farmaceútico. El epidemiólogo será el proceso principal y creará al farmacéutico y al médico. El médico a su vez a crear a tantos procesos hijos como pacientes se especifiquen como argumento en el script. Una vez comience el programa y se creen los hijos el médico quedará a la espera de que el farmacéutico le proporcione las vacunas, cuando se disponga de ellas el médico podrá comenzar a vacunar a los pacientes que deben quedarse esperando.

Para simular esta situación el epidemiólogo (proceso principal) una vez creado los procesos hijos va a esperar 2 segundos y enviará una señal SIGUSR1 al proceso farmacéutico, este calculará un aleatorio entre 0 y 1. Si es cero no hay dosis suficientes y devuelve ese valor a su proceso padre, si es uno, sí hay dosis y también se lo comunicará. En caso de que no haya dosis el proceso epidemiólogo aborta la ejecución del programa. Si hay dosis suficientes el padre mandará la señal SIGUSR2 al médico, este comenzará la vacunación. Para ello manda secuencialmente la señal SIGUSR1 a cada uno de los pacientes que estaban esperando, esto supone que vacuna a uno, espera resultado y hasta que no le devuelve un valor no vacuna el siguiente. El tendrá que esperar a que el médico le vacune, dormir 2 segundos y calcular un número aleatorio entre 1 y 10 que devuelve al médico. Si el valor devuelto es par tiene reacción, sino no la tendría. El médico deberá devolver al epidemiólogo la información del número de pacientes que han tenido reacción.

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