Práctica: Sincronización de procesos

Miguel Ángel Conde González
Antonio Gómez García
Luis Panizo Alonso

November 5, 2018

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 simula una carrera de karts. 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 corredores que participa en la carrera y que luego se pasara como argumentos al programa.
- En caso de que se escoja la cuarta, se saldrá del script.

Carrera de karts

El programa va a simular una carrera de karts. En dicha carrera existe un juez principal, un juez de pista y una serie de corredores. El juez principal será el proceso principal, el juez de pista será un proceso hijo de éste y los corredores serán también procesos hijos del principal.

El número de corredores vendrá dado como argumento al programa. Si se recibe como argumento un 3 se crearán 3 procesos hijos que los simulen.

El juez principal (proceso principal) al lanzarse va a crear tanto el juez de pista como los corredores, una vez creados deben quedarse esperando la recepción de una señal.

Para comenzar la carrera el juez principal debe asegurarse de que la pista está en condiciones. Para ello va a enviar un señal SIGUSR1 al juez de pista. El juez de pista verifica el estado de la misma. Para comprobar esto, dicho proceso domirá 5 segundos y generará un aleatorio entre 0 y 1. Si se trata de un 0 manda una señal SIGUSR1 al proceso principal para comunicarle que la carrera NO puede continuar, en caso de que SÍ se pueda correr se manda una señal SIGUSR2 y el juez de pista finalizaría su ejecución. Si el proceso principal recibe una señal de que no se puede comenzar, duerme 3 segundos y envía una señal al juez de pista para que lo vuelva comprobar, para lo que se procedería de la forma ya comentada con anterioridad.

Una vez el juez principal recibe que la pista está lista mandará una señal a los corredores y se quedará esperando por ellos. En cada proceso hijo se duermen un número aleatorio de segundos entre 1 y 5 y se devuelve al proceso padre el número de segundos que se ha dormido. Una vez hecho esto el hijo finaliza la carrera.

El proceso principal debe mostrar la lista de tiempos de los procesos que corren, determinando PID y número de segundos empleados en la carrera. Cuando se haya impreso esto se acabara la carrera y se imprimirá un mensaje al respecto.

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