Escuela Politécnica Superior de Elche
Grado en Ingeniería Informática en Tecnologías de la Información

Sistemas Operativos

Práctica
Programación Shell Script y C

Objetivos

El objetivo de esta práctica es familiarizar al alumno con la programación concurrente básica en Ansi C sobre plataformas Unix-Like y su integración con la programación Shell Script de estos sistemas.

Descripción general de la práctica

La práctica consiste en la creación de un generador de sorteos de una lotería 6/15 con informes de resultados que tiene dos partes diferenciadas, aunque integradas en una única solución común. La parte de Shell Script y la parte de C.

La parte de Shell Script consistirá en un programa al que se le pasarán como parametros el número de sorteos y el número de jugadores. Con la sintaxis del siguiente ejemplo:

```
$ loteria 15 10
```

Donde 15 sería el número de sorteos y 10 el número de jugadores.

El programa **loteria** (Shell Script) lanzará 15 veces al programa **sorteo** (en C) al que le pasará en cada llamada el número de sorteo de que se trate y el número de jugadores que recibe como parámetro.

El programa **sorteo** (en C) creará tantos hijos como se indique (jugadores) que comprobarán su apuesta con la combinación generada por el padre. Es decir el programa **sorteo** tendrá un código para el padre y un código para el hijo. El padre genera una combinación ganadora y los hijos generan una combinación cada uno. Cada hijo comparará su combinación con la del padre y contestará al padre diciéndole cuantos

aciertos tiene. El padre entonces genera un fichero resultado del sorteo indicando el premio correspondiente a cada jugador en función de los aciertos reportados.

La parte Shell Script, (**loteria**) leerá todos los ficheros de resultado para ir sumando para cada jugador los premios recibidos en cada sorteo, con lo que confeccionará un informe de ganancias que mostrará por pantalla.

Hay que escribir por tanto dos programas, **loteria** y **sorteo**, en Bash y C respectivamente. El programa **lotería** llamará *n* veces al programa **sorteo** tras las cuales leerá los ficheros generados por éste para confeccionar un resultado que mostrará por pantalla.

La comunicación entre los padres e hijos del programa **sorteo** será mediante tuberías (pipes) y mediante el valor de terminación del hijo (exit), es decir el padre genera la combinación ganadora y la volcará en un pipe. El hijo generará una combinación aleatoria y la comparará con la del padre que la lee del pipe. Con esto determinará cuantos aciertos tiene, informando al padre mediante la función de terminación *exit(aciertos)*.

La comunicación entre el shell script **loteria** y el programa C **sorteo** será mediante fichero. El programa **sorteo** generará un fichero con el resultado en premios del sorteo, que leerá el programa **loteria** una vez terminados todos los sorteos.

El nombre de los ficheros indica su contenido con la siguiente sintaxis (ver ejemplo):

`SnR` Será el nombre del fichero de resultados del sorteo número n

Ejemplo: S1R (Resultados del sorteo 1 para 10 jugadores, uno por línea)

```
#Resultados Sorteo 1
0
10
0
50
500
0
10000
0
0
0
```

La primera línea es un comentario que se puede omitir. El programa lotería deberá funcionar correctamente tanto si el fichero tiene el comentario como si no.

Cada línea es el importe en premios de un jugador, línea 1 jugador 1, línea 2 jugador 2, etc… Las líneas a cero (0) indican que el jugador no tiene premio en el sorteo (no ha acertado al menos 3 números, ver más adelante la tabla de premios). El jugador 2 tiene 3 aciertos (10€), el jugador 4 tiene 4 aciertos, el 5 5 aciertos y el 7, 6 aciertos.

El programa **loteria** abrirá el fichero de resultados de cada sorteo y sumará acumulando los premios de cada jugador para finalmente informar por pantalla de los premios totales tras todos los sorteos. Puede leerlos todos al final o conforme se van realizando los sorteos, como se quiera. Cuando se vuelva a llamar al programa **loteria** primero eliminará los ficheros de resultados de la ejecución anterior si los hubiera.

Desarrollo de la práctica.
Descripción detallada y especificaciones

Programa sorteo :

El proceso padre:

- Llevará el control del pid y el número de orden de cada hijo en un array de estructuras HIJO. Ver más adelante.
- Recibirá como parámetro dos valores el número de sorteo y el número de hijos (jugadores) que participan en el sorteo.
- Realizará la validación oportuna de los parámetros de tal forma que:
  - El número de sorteos puede ser cualquier numero del rango 1..15
  - El número de jugadores puede ser cualquier número del rango 1..10
  - Los parámetros son obligatorios.
  - Cualquier error de parámetros provocara la terminación del programa devolviendo -1 (e informando del error por pantalla), lo que será tenido en cuenta por el programa **loteria.**
- El padre creará tantos procesos hijos como se indiquen, y registrará el pid de cada hijo en el array en su estructura correspondiente.
- **El padre no esperará la terminación del hijo para crear otro hijo**.
- Cuando el padre haya terminado de crear los hijos **esperará la confirmación de apuesta de todos los hijos**.
- **Sincronización 1:**
  - **Sincronización entre padre e hijos mediante señales (Opción 1)**
    - Cada hijo enviará una señal al padre indicando que ha realizado su apuesta.
  - **Sincronización entre padre e hijos mediante semáforos (Opción 2)**
    - El padre creará un conjunto de semáforos, uno para cada hijo y uno para el mismo.
    - El mecanismo de sincronización será una barrera múltiple donde el padre actúa como controlador.
- Cuando todos los hijos hayan confirmado que ha hecho su apuesta mediante uno de los mecanismos propuestos (opción 1 u opción 2), el padre generará la combinación ganadora y la volcará en el pipe **habiendo sido ordenada de menor a mayor** previamente y siguiendo las siguientes reglas:
  - La combinación ganadora será una combinación aleatoriamente generada de 6 números elegidos entre los 15 primeros números, es decir los números a los que se puede jugar están en el rango 1..15
  - La combinación ganadora no puede tener números repetidos
  - No hay complementario ni reintegro. Solo 6 números de entre 15.
- **Sincronización 2:**
  - **Sincronización mediante señales:**
    - El padre **enviará una señal** a todos sus hijos indicando que la combinación ganadora ha sido generada.
    - Una vez recibida la señal cada hijo puede leer del pipe.
  - **Sincronización mediante lectura bloqueante del pipe:**
    - La simple lectura del pipe que comparte con el padre bloquea al hijo caso de que el padre no haya escrito en el pipe.

- El padre **esperará la terminación de todos los hijos**. En el estado de terminación de los hijos vendrá el número de aciertos de cada uno. Este dato también **lo registrará el padre en la estructura del hijo correspondiente** dentro del array.
- El padre **generará el fichero de resultados** con la sintaxis especificada y con la siguiente tabla de premios.

    - Menos de tres aciertos premio 0€
    - Tres aciertos premio 10€
    - Cuatro aciertos premio 50€
    - Cinco aciertos premio 500€
    - Seis aciertos premio 10.000€

Los hijos:

- Lo primero que realiza el hijo es la apuesta, siguiendo **las mismas reglas** citadas anteriormente para la combinación ganadora.
- Sincronización 1: El hijo **notificará al padre mediante el envío de una señal** que ya ha realizado su apuesta (opción 1) o se parará en la barrera definida (opción 2) hasta que el controlador (padre) indique que puede seguir (cuando todos los hijos hayan llegado a la barrera).
- Sincronización 2: El hijo **esperará la llegada de una señal** del padre indicando que ya se ha generado la combinación ganadora y se pondrá a leer del pipe (opción 1), o directamente se pondrá a leer del pipe (opción 2)
- El hijo **leerá del pipe** la combinación ganadora y comprobará cuántos números ha acertado.
- El hijo terminará su ejecución **informando al padre en su estado de terminación** de cuantos aciertos ha tenido.

Programa loteria :

Será el encargado de lanzar la ejecución del programa **sorteo**.

- Recibirá como parámetros el número de sorteos a realizar y el número de jugadores. Los mismos jugadores para todos los sorteos. Por ejemplo:
    - `$ loteria 15 10`
    - Se realizan 15 sorteos de 10 jugadores cada sorteo.
- Realizará el control de los parámetros, de forma que mostrará un error si los rangos no se ajustan a lo especificado para el programa **sorteo** o no se pasa alguno de ellos puesto que son obligatorios.
- En caso de error mostrará la ayuda de utilización del programa.
- Analizará el directorio de trabajo para determinar si existen ficheros de resultado de una ejecución anterior, en cuyo caso procederá a borrarlos.
- Realizará un bucle para llamar tantas veces al programa **sorteo** como números de sorteos se indiquen como parámetro, pasando cada vez el número de sorteo correspondiente y el número de jugadores al programa **sorteo** según la sintaxis de este.
- Tras la ejecución de cada sorteo en el directorio de trabajo se encontrará el fichero de resultados.
- Abrirá uno a uno todos los ficheros de resultados e irá llevando la suma de las ganancias de cada jugador.

- Presentará un informe de resultados con el siguiente formato:

```
Informe de resultados
Numero de sorteos: 15   Numero de jugadores: 10
Jugador 1: Total premio    nnnn Euros.
Jugador 2: Total premio    nnnn Euros.
…….
Jugador 10: Total premio  nnnn Euros.
```

Requisitos

- Hacer un **código modular y documentado** :
  - main.c : Incluirá únicamente la función main que llamará al resto de funciones
  - func.c : Tiene las funciones, con cabecera
  - func.h : Prototipos y constantes
- Utilizar **make** para compilar todo el proyecto utilizando el makefile correspondiente.
- Eliminar correctamente los ficheros temporales
- Cerrar correctamente los pipes
- Procesos hijos deben terminar correctamente y no después que el padre.
- Generar funciones robustas a los posibles fallos en su funcionamiento.
- El código fuente debe estar comentado adecuadamente. Ni excesos ni defecto.

NOTA: El incumplimiento de estos requisitos restará un punto por cada uno no cumplido.

Estructura HIJO. Puede ser ampliada y modificada convenientemente.

```
typedef struct
{
  int pid ;      //pid del hijo
  int num ;      //numero de orden del hijo
  long premio;  //premio del hijo en un sorteo
} HIJO ;
```

Entrega

- La práctica se realizará de manera individual o por parejas.
- El plazo límite de entrega será **hasta las 24h del día del examen de teoría.**
- La entrega se realizará mediante e-mail al profesor (mmrach@umh.es)
- En el asunto del correo electrónico debe aparecer el siguiente texto:
  SO PRACTICA DE <nombre_y_apellidos_del_alumno/alumnos>

Se entregará un fichero comprimido (.zip o .rar) con:

1. La memoria de la práctica
2. El código fuente debidamente documentado

La memoria se entregará en formato **.doc** o .**pdf.**
El formato es libre pero hay que cuidar la presentación de la memoria.

Dicha memoria deberá incluir la siguiente información, siendo todos los apartados objetos de evaluación. La entrega de la memoria con todos sus apartados es requisito indispensable para la corrección de la práctica.

- **Nombre y DNI del alumno/alumnos**
- **Consideraciones de diseño: (Cuerpo principal de la memoria, organizado según convenga)**
  En este apartado se detallarán las técnicas y estrategias utilizadas para realizar la práctica, enumerando los componentes del lenguaje que se han utilizado. Se deberá defender los criterios y decisiones que se han tomado a la hora de decidir por una implementación u otra, es decir, se deberán comentar y justificar las decisiones de diseño que se han tomado durante el desarrollo de la práctica. Para favorecer la explicación de cada técnica o estrategia se puede mostrar el código fuente afectado.
- **Consideraciones finales del alumno**
  En este apartado el/los alumno/s explicarán a modo de reflexión personal (o del grupo) sobre el proceso llevado a cabo, todas aquellas consideraciones que crean oportunas. Se comentarán aspectos como cuáles han sido los puntos más difíciles, cuales los más sencillos, reflexión sobre la relación tiempo dedicado vs. dificultad, y sobre lo que se ha aprendido, así como posibles mejoras que a su juicio se pudieran aplicar al programa.