Proyecto I: juego de dados (15%)

Abril-Julio 2013

Versión: 28 de mayo de 2013

1 Objetivos

- Familiarizarse con los mecanismos de creación y manejo de procesos e hilos de ejecución en GNU/Linux
- Familiarizarse con mecanismos básicos de comunicación entre procesos, y entre hilos

2 Enunciado

Se quiere simular un juego de dados entre varios jugadores. El juego consiste en que cada uno lanza los dados (son dos dados) un número de veces determinado previamente (e igual para todos). Ganará el que totalice mayor número sumando todas las tiradas.

Se harán dos versiones del programa, una usando procesos (`juegodedados_p`) y otra usando hilos (`juegodedados_h`). Cada jugador será representado por un proceso (o hilo), y habrá un coordinador (el *croupier*) que se encarga de darle los dados (en realidad, la semilla para la generación de números aleatorios) a cada jugador, y de determinar quién es el ganador.

2.1 Sintaxis

Ejecución de la versión procesos

```
> juegodedados_p [-n i] [-s x] [-j y] [-h]
```

Ejecución de la versión hilos

```
> juegodedados_h [-n i] [-s x] [-j y] [-h]
```

- "`-n i`" indica el número de tiradas (`i`) que hará cada jugador. Por defecto, se usará el valor 10. El máximo es 20. Esto debe verificarse y dar un error si se especifica un número de tiradas mayor que 20.

- "`-s x`", el argumento `x` es un entero que corresponde a la *semilla* del generador de números aleatorios que utilizará el *croupier* (ver abajo). Por defecto, la semilla tiene el valor 1.

- "`-j y`" indica el número de jugadores. Esta opción es obligatoria, salvo si se usa la opción `-h`. No hay valor por defecto.

- "`-h`" corresponde a la ayuda (*help*). Imprime un mensaje de ayuda y termina. El mensaje de ayuda es del estilo del mensaje de un programa utilitario de Unix invocado con la opción -h. Consiste en una breve explicación de lo que hace el comando y la sintaxis (entradas, opciones y parámetros de cada opción).

Notas:

1. el programa debe aceptar las opciones en cualquier orden; por ejemplo, primero `-n i` y luego `-s x`, o al revés.

2. Las opciones `-n`, `-j` y `-s` deben estar siempre seguidas por un entero.

3. Se sigue la convención en Unix para los parámetros de línea de comandos: parámetros (con o sin argumentos) encerrados entre corchetes significa que pueden o no estar presentes en la línea de comandos.

2.2 Semántica

El programa se inicia imprimiendo por pantalla el valor de los argumentos pasados en la línea de parámetros. El *croupier* genera aleatoriamente la semilla para cada jugador usando la función `rand()` y la semilla pasada como argumento. Cada jugador tiene un identificador, comenzando por 1, hasta el número de jugadores.

El *croupier* genera un proceso o hilo por cada jugador y le pasa su semilla usando una variable. Cada jugador escribe en un archivo llamado "`tiradas_i`", donde `i` es su identificador, la secuencia de números que generó. Al final del archivo, debe imprimir el total acumulado.

El resultado (total) de cada jugador es pasado al *croupier* a través del código de estatus, retornado en el argumento de la función `exit`, en el caso de procesos, y `pthread_exit`, en el caso de hilos.

El *croupier*, una vez que todos los jugadores terminaron, imprime por pantalla el indentificador del proceso o hilo que ganó, así como la puntuación ganadora.

2.3 Salida

En los ejemplos mostrados a continuación se usó "juegodedados_p". Debe entenderse que la salida es exactamente igual para ambas versiones (hilos y procesos). El siguiente es un ejemplo de lo que debe verse por la salida estándar

```
> juegodedados_p -n 5 -s 2000 -j 4
Numero de tiradas: 5
Semilla: 2000
Jugadores: 4
**** Gana el jugador 2, con 48 puntos
```

Esta corrida genera cuatro archivos, uno por jugador. Este es un ejemplo del archivo generado

```
Jugador: 2
Numero de tiradas: 5
Tirada 1: 12
Tirada 2: 8
Tirada 3: 7
Tirada 4: 3
Tirada 5: 9
TOTAL:   39
```

Opción de ayuda:

```
> juegodedados_p -h
juegodedados_p: imprime una secuencia de numeros aleatorios
SINTAXIS:
juegodedados_p [-n i] [-s x] [-h]
 -n i: tamano de la secuencia (por defecto, 10)
 -s x: semilla (por defecto, 1)
 -j y: numero de jugadores
 -h: imprime esta ayuda y sale
```

3 Observaciones importantes

1. PROYECTO QUE NO CORRA O NO COMPILE NO SERA CORREGIDO. Los programas deben poder compilarse y ejecutarse en cualquiera de las estaciones (GNU/Linux) del LDC. Si Ud. realizó el programa en su casa o en la USB, pero en alguna otra plataforma, debe asegurarse antes de la entrega que su proyecto funcione en las estaciones GNU/Linux antes mencionadas.
2. Es obligatorio usar las funciones `rand` y `srand`.
3. Se debe chequear el valor de retorno de las llamadas al sistema.
4. Deben hacer uso del makefile.
5. Deben hacer un programa modular, legible y documentado.
6. Deben respetar los formatos de salida especificados aquí. Este enunciado debe tomarse como un contrato, y las violaciones serán penalizadas.