Trabajo práctico de Sistemas Operativos I

1er cuatrimestre de 2010

Adivinar el número correcto usando IPC

Escribir una aplicación cliente/servidor en Linux, que use mensajes entre procesos (IPC) que permita que varios jugadores adivinen el número correcto que genera un servidor.

Las reglas del juego son:

1. El juego se juega de a dos jugadores a la vez, uno que genera el número (generador) y el otro que lo adivina (adivinador).
2. El generador elige un número, (en formato string), de 4 dígitos, todos distintos, que es el código. El orden de los números en el código es importante: códigos con números iguales, en distinto orden, son distintos. Los dígitos posibles son 0, 1, 2, 3, 4, 5, 6, 7, 8 y 9.
3. El adivinador debe adivinar el código luego de 5 intentos. Cada intento consiste en una comunicación con el servidor, donde se envían 4 dígitos.
4. El generador responde a cada intento con dos números B y R (bien y regular). Estos números indican cuántas dígitos están correctos y en la posición correcta (B) y cuántas son correctas, pero están en posición errónea (R). Por ejemplo, si el código fuera ``1234", la respuesta al intento ``*7314"* será *(B,R)=(2,1),* porque los dígitos `3' y `1' están en el código, pero en distintas posiciones, y el `4' está en la posición correcta.

   Nota: Es obvio que ***B+R <= 4****.*

5. El adivinador puede usar la respuesta para generar un nuevo intento siguiendo alguna estrategia.
6. El generador gana si el adivinador no logra adivinar el código luego de sus cinco intentos.

   En la aplicación el servidor juega el papel del ***generador*** para varios adivinadores a la vez, que juegan juegos independientes, tal como hace el sistema operativo con los procesos. Elige un código distinto para cada cliente, y le responde por separado y sin equivocar la respuesta.

Las especificaciones del juego son las siguientes:

1. El servidor no es interactivo, solo interactúa con los clientes.
2. La mejor forma de implementar el servidor es con un daemon.
3. Los clientes deben trabajar independientemente de la línea de comandos, o pueden optar por pedir ingreso de códigos por teclado (modo interactivo).
4. En el modo interactivo se le da al usuario un prompt, se lee el código, se le envía el código al servidor y luego se mostrará al cliente la respuesta. Los usuarios pueden comenzar un nuevo juego al concluir el presente, o dejar de jugar.

   Deben usarse estos comandos:

   - **New** – Para salir del juego actual y comenzar uno Nuevo
   - **Quit** – para salir del juego y de la sesión interactiva;
   - **Help** – para pedir ayuda sobre cómo jugar.

   Ante un comando erróneo se muestra la *ayuda*.

   Los comandos no distinguen mayúsculas de minúsculas: **HELP** o **help** es válido por igual.

   En el modo no interactivo los clientes juegan usando una estrategia por programa.

   Puede usarse cualquier estrategia, inclusive la aleatoria, pero el grupo deberá diseñar una propia (y tendrán 10% del valor de la nota en una estrategia bien diseñada, que perderán con la elección de una simple o aleatoria).

5. Un mismo cliente no puede jugar varios juegos a la vez.
6. Los clientes y el servidor intercambian intentos usando mensajes. Pueden usarse colas FCFS en el servidor para implementar las esperas, pero cada grupo puede utilizar su propia estrategia. Los mensajes de los clientes no deben perderse ni desatenderse.
7. Tanto el Servidor como los clientes no pueden ser parientes (padre-hijo o tener un ancestro en común)
8. Se definirá un protocolo para la comunicación, según cómo sea el diseño de cada cliente y de cada servidor. Cada grupo definirá el protocolo y explicará los fundamentos, los comandos y las respuestas esperadas y erróneas.

9. Pueden suponer que los clientes son honestos, y nunca tratarán de jorobar a otro cliente que está jugando a propósito. Si no, puede también implementarse un mecanismo de seguridad.
10. El número total de jugadores que el servidor puede aceptar debe predefinirse, pero debe ser superior a 5 para aprobar el trabajo. Este número debe indicarse al inicializar el servidor.
11. Queda a criterio del grupo decidir el puntaje del juego. Expliquen la forma de obtener puntos y cómo se suman.
12. En caso de caer, la aplicación lo hará con la gracia de una bailarina clásica y no como una bolsa de papas.
13. La aplicación debe ser ``IPC clean'': limpiar las colas de mensajes al salir.

Además se les pide que:

- Discutan los problemas de concurrencia que pueden aparecer entre el servidor y los clientes. Describan el problema y den ejemplos que PUEDAN SUCEDER durante un juego. No inventen cosas extrañas o imposibles.
- Expliquen los algoritmos que usen.
- Documenten el programa debidamente.

Ultimas consideraciones (y muy importantes):

- El programa se escribirá en lenguaje en C y en sistema operativo Linux, cualquier versión de compilador y de sistema operativo. Puede usarse Visual C++ para la interfaz gráfica, pero no es necesario dado que sólo se les pide una línea de prompt.
- La entrega del trabajo incluye la documentación pedida, los fuentes y los programas en ejecución.
- Las fechas de entrega fueron explicitadas en clase, pero recordamos aquí que son: última fecha de clase (con o sin gripe A) y primera fecha de final de julio. Los trabajos entregados más allá de esas dos fechas no serán aceptados sin excepción.