Laboratorio 2:

Profesor: Viktor Tapia
Ayudantes de cátedra: Martín Rodríguez y Tomás Rebolledo
Ayudantes de Tarea: Javiera Cárdenas y Nicolás Toro

10 de Abril 2024

1 Reglas Generales

Para la siguiente tarea se debe realizar un código programado en lenguaje C/C++. Se exigirá que los archivos se presenten de la forma más limpia y legible posible. Deberá incluir un archivo README con las instrucciones de uso de sus programas junto a cualquier indicación que sea necesaria, y un archivo MAKE para poder ejecutar el programa.

2 Enunciado

En la cima de la lucha libre, WrestleMania es el escenario definitivo donde los héroes y las leyendas se encuentran. En esta edición especial, cuatro luchadores legendarios son convocados para competir en el WrestleMania Royale. En el centro del ring, en medio del fervor de los fanáticos, se libra una batalla épica donde solo uno puede reclamar el "Título Supremo" y asegurar su lugar en la historia como el campeón definitivo de WrestleMania.

3 Tarea

Se requiere crear un juego simple similar a un Battle Royale. Se solicita desarrollar el juego de modo que un participante pueda jugar a través de la consola, además de crear tres jugadores extra para que el juego tenga un total de cuatro jugadores. Los jugadores adicionales deben ser generados mediante la función fork() y utilizando pipes para la transmisión de información.

3.1 Conceptos básicos

Se establecen los conceptos básicos del juego. Cualquier modificación a estos conceptos debe ser detallada en el README entregado junto con el programa:

1. Hay cuatro jugadores, cada uno con características específicas:
    - Vida: 100 puntos.
    - Ataque: valor aleatorio entre 30 y 40.
    - Defensa: valor aleatorio entre 10 y 25.
    - Evasión: 60 - Defensa.
2. El objetivo del juego es que quede un único jugador en pie, donde:
    - Todos los jugadores pueden atacar a cualquier otro durante la ronda.

- Los jugadores tienen la posibilidad de esquivar ataques en función de su Evasión (por ejemplo, si la Evasión es 40, tienen un 40% de probabilidad de esquivar).
- Si un jugador no es capaz de esquivar un ataque, el daño infligido se calcula como

$$\text{Daño} = \text{Ataque - Defensa}$$

y se resta de su Vida actual.

3.2 Inicio de la partida

1. Se calculan los valores de Vida, Ataque, Defensa y Evasión de cada jugador.
2. Se muestran por consola los valores calculados para cada jugador.
3. Comienza el juego.

3.3 Desarrollo de la Partida

La partida se desarrolla por rondas. Cada ronda sigue el siguiente proceso:

1. Se muestra la vida actual de todos los jugadores. El jugador controlado por consola elige a quién atacar, mientras que los otros procesos o jugadores lo hacen de forma aleatoria.
2. Se muestra quiénes atacan a cada jugador, seguido de quiénes lograron esquivar los ataques.
3. Se realiza la reducción de la vida de cada jugador según los daños recibidos; si no recibió daño, no se reduce la vida.

3.4 Fin de la Partida

La partida concluye cuando solo queda un jugador en pie con vida mayor a 0. En caso de empate, se declara un ganador aleatorio.

4 Jugadores y procesos

El juego consta de 4 jugadores, siendo uno de ellos controlado por usted. Cada jugador se representa como un proceso separado, lo que significa que su programa ejecutará un total de 5 procesos (uno padre y 4 hijos). Cada jugador tiene la tarea de seleccionar a quién atacar y calcular si logra esquivar los ataques recibidos durante una ronda. El orden de los jugadores es primero el controlado por consola, seguido de los otros 3. Una vez que un jugador gana, el proceso padre debe indicar cuál de ellos fue el ganador, notificar a los procesos hijos y luego cerrar todos los procesos. Es importante recordar que para terminar el programa, los procesos hijos deben finalizar primero y luego el proceso padre.

4.1 Visualización en la Consola

El diseño del formato de visualización del juego queda a su creatividad. Sin embargo, se requiere mostrar al menos los siguientes elementos:

- Características de cada jugador: Al inicio de la ronda, se deben mostrar la Vida, Ataque, Defensa y Evasión de todos los jugadores.
- Ronda Actual: Se debe indicar en pantalla el número de ronda mediante un mensaje breve.
- Jugadas de todos los jugadores: Al finalizar la ronda, se debe mostrar a quién atacó cada jugador y si su ataque tuvo éxito.

5 Presentación Aleatoria

Para cada tarea, se seleccionarán grupos al azar para presentar su tarea frente a ayudantes y eventualmente profesor, recibiendo una ponderación del 80% y 20% entre tarea y presentación respectivamente. Si su grupo presentó en una tarea, no volverá a salir nuevamente. Se comunicará días antes que grupos presentarán.

6 README

Debe contener como mínimo:

- Nombre, RoI y Paralelo de los integrantes.
- Especificación de los nombres de los archivos.
- Instrucciones generales de compilación y uso.

7 Consideraciones Generales

- Se deberá trabajar de a pares. Se deberá entregar en sus repositorios de Github a mas tardar el día 8 de Mayo de 2024 a las 23:59 horas. Se descontarán 10 puntos por cada hora o fracción de atraso. Las copias serán evaluadas con nota 0 en el promedio de las tareas.
- La tarea debe ser hecha en el lenguaje C/C++. Se asume que usted sabe programar en este lenguaje, ha tenido vivencias con el, o que aprende con rapidez.
- Pueden crear todas las funciones auxiliares que deseen, siempre y cuando estén debidamente comentadas.
- Las tareas serán ejecutadas en **Linux**, cualquier tarea que no se pueda ejecutar en dicho sistema operativo, partirá de nota máxima 60.
- Los archivos deberán colocados en una carpeta dentro de su repositorio llamado "Laboratorio 2".
- Las preguntas deben ser hechas por Aula. De esta forma los demás grupos pueden beneficiarse en base a la pregunta.
- Si no se entrega README o MAKE, o si su programa no funciona, la nota es 0 hasta la recorrección.
- Se descontarán 50 puntos por:
  - Mala implementación del Makefile.
  - No respetar el formato de entrega.
  - Solicitar edición de código para el correcto funcionamiento del Laboratorio.