Laboratorio 4: Sistemas Operativos

Profesor: Viktor Tapia
Ayudantes: Nicolás Schiaffino y Tomás Rebolledo
Ayudante de Tarea: Javiera Cárdenas y Nicolás Toro

Noviembre 2024

1 Reglas Generales

Para este laboratorio, el código debe ser programado en Python (versión 3 o superior). Los archivos deberán presentarse de manera clara y legible, incluyendo un archivo README con instrucciones de uso y cualquier otra información relevante para su ejecución. Para esta tarea no se requiere Makefile.

2 Contexto

En los rincones más vibrantes de la Chilean Premier League, 256 jugadores estrella se han reunido para competir en el torneo más esperado del año: el Torneo "Experto Easy". Solo uno de estos talentosos deportistas podrá consagrarse como el "Jugador Experto Easy", ganando no solo la gloria, sino el prestigio de ser coronado como el mejor de toda la liga.

3 Enunciado

El objetivo de esta tarea es diseñar un sistema que gestione un torneo donde cada competidor es representado por una hebra. Este torneo incluye una fase de validación inicial, una fase de eliminación directa y una fase de repechaje, donde las hebras eliminadas en cada ronda tienen una segunda oportunidad para regresar al torneo y avanzar hacia la final.

4 Estructura del Torneo

El torneo tiene tres fases:

- **Fase de Validación Inicial:** Los 256 jugadores llegan para demostrar sus habilidades en una prueba de validación. Aquí, cada jugador se prepara con desafíos intensos que ponen a prueba su agilidad, precisión y dominio técnico. Cada zona de validación tiene capacidad para evaluar a un grupo de 32 jugadores a la vez. Una vez validados, los jugadores avanzan a la fase de eliminación directa.
- **Fase de Eliminación Directa (Ganadores):**Los jugadores validados pasan a enfrentarse en duelos uno contra uno, donde cada enfrentamiento define quién seguirá avanzando en el torneo . Solo los ganadores de cada duelo siguen adelante en la competencia, mientras que los perdedores reciben una última oportunidad en la "Fase de Repechaje".

- **Fase de Repechaje (Perdedores):** Los jugadores eliminados luchan para obtener un segundo intento. Aquí, se enfrentan nuevamente en pares en duelos a muerte futbolística para lograr volver al camino de la victoria. Solo uno logrará sobrevivir a esta batalla, ganando el derecho de desafiar al finalista invicto.

5 Detalles de Implementación

- **Número de Hebras:** 256
- **Duración de validación:** 15 segundos
- **Duración de cada enfrentamiento:** 10 segundos.

6 Organización

Al iniciar el programa, se generarán 256 hebras que representarán a los jugadores. Cada hebra comenzará en la fase de validación, pasando en en grupos de 32 a una de las dos zonas de validación disponibles. Tras superar la validación, las hebras entrarán en la fase de eliminación directa.

En la fase de eliminación directa, las hebras compiten en enfrentamientos por parejas. Los ganadores avanzan, mientras que los perdedores pasan a la fase de repechaje. La fase de repechaje permitirá que los perdedores continúen compitiendo hasta llegar a una final entre el ganador de la fase de eliminación directa y el ganador de la fase de repechaje.

7 Registros

El programa debe generar archivos de texto para registrar el progreso de cada hebra en el torneo:

- `Validación.txt`: Registro de la fase de validación inicial. Cada línea debe incluir el identificador cada hebra, el tiempo de entrada a la zona de validación y el resultado de la validación. Ejemplo: `Hebra1, Hebra2, ..., Hebra32 14:10:47.271243, Validación Completa`
- `Ganadores_Ronda1.txt`, `Ganadores_Ronda2.txt`, etc.: Registros de las rondas en la fase de eliminación directa, indicando los enfrentamientos y resultados. Ejemplo: `Hebra1 vs Hebra2 14:11:02.271243, Ganador Hebra2`
- `Perdedores_Ronda1.txt`, `Perdedores_Ronda2.txt`, etc.: Registros de las rondas en la fase de repechaje. Ejemplo: `Hebra1 vs Hebra4 14:11:15.271243, Ganador Hebra1`
- `Final.txt`: Registro final con el identificador de los dos finalistas (ganador de la fase de eliminación directa y ganador de la fase de repechaje) y el resultado final del torneo. Ejemplo: `Hebra2 vs Hebra4 14:11:30.271243, Ganador Hebra2`

8 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 qué grupos presentarán.

9 README

El README debe incluir:

- Nombres y roles de los integrantes.
- Especificación de los nombres de archivos de registro.
- Instrucciones para ejecutar el programa.

10 Consideraciones Generales

- Todas las hebras deben terminar su ejecución antes de que el programa concluya.
- Para implementan las hebras, locks y semaforos deben utilizar la libreria Threading
- Se deberá trabajar de a pares. Se deberá entregar en Github a más tardar el día 24 de Noviembre 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 Python. Se asume que usted sabe programar en este lenguaje, ha tenido vivencias con él, 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.
- Las preguntas deben ser hechas por Aula o por Correo. De esta forma, los demás grupos pueden beneficiarse en base a la pregunta.
- Si no se entrega README, o si su programa no funciona, la nota es 0 hasta la recorrección.
- Para esta entrega no se necesita un Makefile.