PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2233 - Programación Avanzada
1^{er} semestre 2017

Actividad 10

Threads

Introducción

El "benevolente" Doctor Mavrakis ha decidido regalar como promedio final del curso un siete a 3 personas que estén cursando IIC2233-3/4. Para evitar arbitrariedades, ha creado el Torneo de los Tres Programadores que consta de una única prueba: sobrevivir al laberinto.

Instrucciones

En esta actividad debes usar tus conocimientos de *threading* para modelar el funcionamiento de la competencia. Las reglas de la competencia son simples: Cuando tres personas lleguen al final del laberinto, la competencia se acaba. Para mantener la realidad de la competencia, cada persona deberá ser independiente de las demás, por lo que un requisito es que sean implementadas como *threads*.

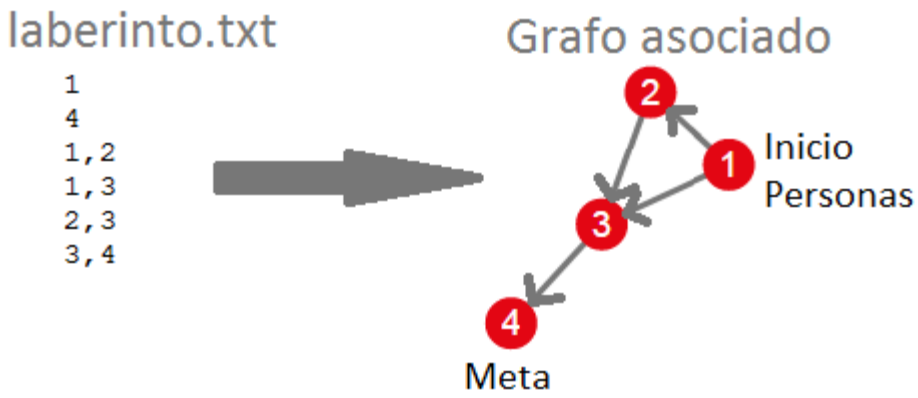
Ojo: No se pide que las personas crucen el laberinto de manera inteligente, solo que lo hagan por los caminos que encuentren disponibles (los que logren salir del laberinto solo tendrán suerte).

La simulación deberá tener las siguientes clases:

- `Laberinto`: Es un grafo direccional, en donde cada nodo del grafo será una pieza del laberinto. Cada pieza es bastante pequeña, por lo que solo puede haber 1 persona a la vez (si hay una persona en la pieza A, y una nueva persona intenta entrar, esta deberá esperar a que la pieza se desocupe antes de entrar), y el tiempo de estadía en una pieza es un número aleatorio entre 1 y 3 segundos (es decir, al entrar a una pieza, esta deberá quedar cerrada durante 1, 2 o 3 segundos, y una vez terminado este tiempo, la persona que estaba adentro se va a otra pieza, y deja la pieza libre para que otros puedan entrar).

  Para crear el grafo, se te entrega el archivo `laberinto.txt`, que incluye las especificaciones del laberinto. La primera línea, es un número que indica el id de la pieza de inicio. La segunda línea, es el id de la meta, y el resto de las líneas son de la forma num1,num2, lo que significa que uno puede ir de la pieza num1 a la pieza num2, pero no necesariamente de la num2 a la num1.

  A modo de ejemplo, puedes ver la siguiente imagen para ver cómo se debería armar un laberinto, dado un archivo txt en particular.



Para agregarle dificulad a la prueba, el Doctor Mavrakis libera una toxina en el laberinto, que le quitará vida a los participantes cada 1 segundo (la cantidad de vida perdida se indica más adelante).

- `Persona`: Serán los *threads* que se van moviendo por el laberinto. Estos poseen los siguientes atributos y métodos:
  - `hp`: Número aleatorio entre 80 y 120 que indica la resistencia que tendrán las personas durante su estadía en el laberinto. Si llega a 0, la persona muere, por lo que el *thread* desaparece.
  - `pieza_actual`: Contiene el id (o la pieza) en la que se está actualmente.
  - `resistance`: Es un valor aleatorio entre 1 y 3, e indica la resistencia que tendrá a la toxina. La vida que se pierde por la toxina será 6 - `resistance`.

Para la llegada de las personas, debes tener un *thread* corriendo, que se encarga de crear a las personas (una persona recién creada parte en la pieza de inicio). La tasa de aparición de las personas sigue una distribución exponencial con una tasa de llegada de 1 persona cada 5 segundos.

Limpieza

Como a nadie le gusta mantener un espacio sucio y maloliente, el Doctor Mavrakis contrató un barrendero que tendrá como objetivo ir removiendo del laberinto todos los cadáveres. Para implementarlo, se pide que hagas un thread, que se encargará de ir sacando del laberinto a las personas que ya hayan muerto (ie, eliminarlas de la lista de personas).
Ojo: El barrendero no puede retirar un cadaver del laberinto, si su tiempo de defunción no ha sido registrado para las estadísticas finales.

Registro

Durante la ejecución de tu programa, debes ir mostrando la información de los sucesos que van ocurriendo, esto es, informar cuando una persona pierde vida, muere o llega a la meta. En caso de llegar a la meta, se debe informar cuánto tiempo le tomó finalizar el laberinto.

Como al Doctor Mavrakis le interesa ver cómo estuvo su competencia para decidir si se repite en el futuro, se pide que al finalizar la competencia crees un archivo llamado `registro_sucesos.txt`. En este archivo debes registrar las 3 personas que lograron llegar a la meta (id o nombre, junto con el tiempo en que llegó a la meta, y el tiempo que se tardó en finalizar), cuántas personas murieron en el intento, y cuál fue el tiempo promedio de vida antes de morir en el laberinto (cuánto tiempo sobrevivieron en el laberinto, sin considerar a los ganadores). A continuación, se muestra un ejemplo de output:

```
Persona0 - 13.079748153686523 - 13.079748153686523
Persona1 - 18.09503483772278 - 12.074690580368042
Persona3 - 39.25724530220032 - 21.16221046447754
Muertos: 1
Promedio tiempos supervivencia: 20.21415615081787
```

En este ejemplo, la persona1 llegó en el segundo 18 de la simulación a la meta, pero se tardó solo 12 segundos en terminar (es decir, ingresó al laberinto al segundo 6 de la siulacion)

Requerimientos

- (1.00 pts) *Thread* para crear a las personas.
- (1.00 pts) *Thread* encargado de retirar gente fallecida D:
- (3.50 pts) Clase `Persona`.
  - (1.00 pts) Correcto movimiento de las personas dentro del laberinto.
  - (1.00 pts) Correcta escritura de llegadas y muertes.
  - (1.00 pts) Se pierde vida cada 1 segundo.
  - (0.50 pts) Imprimir en pantalla registro de sucesos.
- (0.50 pts) Escritura del archivo.

Notas

- Recuerda que para controlar el uso de recursos simultáneamente puedes usar `Lock` de `threading`.

Entrega

- **Lugar:** GIT - Carpeta: Actividades/AC10
- **Hora:** 16:55