Segundo Proyecto: Concu-mining

75.59 - Técnicas de Programación Concurrente I

Objetivo

Implementar una simulación de una actividad exploratoria de un grupo de trabajadores de la minería.

Requerimientos Funcionales

Un grupo de mineros es contratado para trabajar en un área que no había sido explorada hasta el momento. Disponen de un mapa, un poco borroso y con poca información, conteniendo lo poco que se conoce de la región hasta el momento.

Los mineros son organizados por un *líder* que les da indicaciones sobre las tareas que deben realizar. Todos respetan su palabra.

Para darle más incentivo a los trabajadores, el líder decidió implementar un sistema de premios y castigos para los empleados, organizando el trabajo de la siguiente forma:

1. Comienza la jornada de trabajo: el líder divide la región a explorar en N porciones y le entrega el mapa a cada minero.
2. Los mineros salen a explorar con su carretilla la primera porción de la región en busca de pepitas de oro.
3. Cuando el líder da la voz de regresar, todos los mineros vuelven al punto de encuentro, donde se reúnen en ronda.
4. Por turnos, cada minero revisa su carretilla y canta a viva voz la cantidad de pepitas de oro que encontró en esa exploración.
5. El minero que menos pepitas obtuvo, le entrega sus pepitas al que más obtuvo y se retira del trabajo (castigado por perezoso). Si hay más de un minero con ese número menor, no se retira ninguno y no hay intercambio de pepitas.
6. Cuando termina esta etapa, el líder da la nueva voz de inicio y los mineros salen a explorar una nueva región, repitiéndose los pasos 3 a 6 hasta terminar las N porciones del mapa o hasta que quede un solo minero.

Los siguientes son requerimientos adicionales que deben cumplirse:

- Debe poder configurarse la cantidad de mineros y la cantidad N de porciones de la región.
- Todos los mineros deben escuchar atentamente el canto de los demás.
- Al finalizar la exploración, debe mostrarse la cantidad de pepitas de oro que obtuvo cada minero.

Requerimientos no Funcionales

Los siguientes son los requerimientos no funcionales de la aplicación:

1. El proyecto deberá ser desarrollado en lenguaje **Rust** utilizando Threads y las siguientes herramientas de concurrencia, provistas por el lenguaje y la biblioteca standard *sync*[1]:

   a) Channels (de tipo *multiple producer, single consumer*).
   b) Mutex
   c) Barrier (*https://doc.rust-lang.org/std/sync/struct.Barrier.html*)
   d) Condvar (*https://doc.rust-lang.org/std/sync/struct.Condvar.html*)

   El uso de un módulo externo (crate) debe ser previamente autorizado.

   No se permite el uso de crates de concurrencia.

2. La simulación no deberá tener interfaz gráfica y se ejecutará en una sola consola de línea de comandos.
3. El proyecto deberá funcionar en ambiente Unix / Linux.
4. La aplicación deberá funcionar en una única computadora.
5. El programa deberá poder ejecutarse en "modo debug", lo cual dejará registro de la actividad que realiza en un único archivo de texto para su revisión posterior.

Tareas a Realizar

A continuación se listan las tareas a realizar para completar el desarrollo del proyecto:

1. Dividir el proyecto en *threads*. El objetivo es lograr que la simulación esté conformada por un conjunto de hilos de ejecución que sean lo más sencillos posible.
2. Una vez obtenida la división en *threads*, establecer un esquema de comunicación entre ellos teniendo en cuenta los requerimientos de la aplicación. ¿Qué *threads* se comunican entre sí? ¿Qué datos necesitan compartir para poder trabajar?
3. Realizar la codificación de la aplicación. El código fuente debe estar documentado.

Entrega

La entrega del proyecto comprende lo siguiente:

1. Informe, se deberá presentar impreso en una carpeta o folio y en forma digital (PDF) a través del campus
2. El código fuente de la aplicación, que se entregará únicamente mediante el campus

La entrega en el Classroom estará habilitada hasta las 19 hs de la fecha indicada oportunamente.

El informe a entregar debe contener los siguientes items:

1. Detalle de resolución de la lista de tareas anterior.
2. Diagrama que refleje los threads, el flujo de comunicación entre ellos y los datos que intercambian.
3. Diagramas de entidades realizados.
4. Diagrama de transición de estados de un minero.

[1] https://doc.rust-lang.org/std/sync/index.html