Jose Rubin
Kevin Montesinos

Informe de Proyecto 1.

Introducción

Antes de hablar de cada una de las clases que conforman nuestro proyecto cabe destacar ciertas primitivas que usamos en cada una de las clases para los productores para que no nos diera conflicto el uso de recursos compartidos cada variable volátil tiene su semáforo de tipo Mutex, para que no hayan conflictos entre Threads, también cada producto tiene su almacén en forma de un semáforo normal del tamaño que mande el JSON. Sin más nada que decir explicaremos en el siguiente informe cada una de las clases.

Cada clase que sea un Thread tiene una función que se llama stopRun() que para por completo la ejecución del thread.

Clases

- Productores de todo tipo:
  Primero cada Thread de tipo de productor, este accede al semáforo correspondiente a la cantidad de drive ocupado, si este está lleno, se encola el Thread productor, si no está lleno, pasa al mutex del tipo de productor para tener exclusividad en el drive.

- Assemblers:
  Ambas clases contienen casi el mismo código aunque lo hayamos hecho por separado discutimos ciertos conflictos que nos estaba pasando con los semáforos y las variables que modificaremos. Es importante destacar que para el ensamblador también creamos un semáforo mutex para que solo un ensamblador pueda hacer su trabajo a la vez y no ocurran conflictos. En la función run() del ensamblador, se necesita cumplir una condición (las condiciones dependen de cada serie) si la cumple puede entrar a modificar cada variable pasando por el semáforo mutex de cada una para que no haya conflicto. Si no se cumple la condición el mutex del ensamblador se hace un release() de ese semáforo. En esta función hay un contador de números de teléfonos terminados con su semáforo mutex para que así no haya conflictos en otros lados con esta variable.

- PM: una de las clases más importantes del programa es el PM que se mantiene en constante movimiento entre hacer Sprint Reviews y ver Rick and Morty en cierto intervalo de tiempo dependiendo de en qué productora estamos. El PM es el que mueve las variables del número de días. Primero chequea el número de días si se cumple la condición de que sea mayor a 0, entonces entra en el semaforo Mutex del contador de dias pasa cierta cantidad de tiempo modificara el número de días en cada productora y el del dia actual, setea los textfield del contador, gastos de salario y ganancias. Posteriormente, hay un loop para ver si está viendo Rick y Morty o está haciendo Sprint Reviews. Si no se cumple esta condición todos los hilos mueren.

- Director: otra clase super importante del programa es el director. Al empezar a ejecutarse, chequea el contador si el contador es mayor a 0, en un periodo random de tiempo chequea si está el PM haciendo el sprint review o viendo Rick y Morty para restarle dinero de su salario, el resto del día está trabajando.

- ReadFile: función que retorna la data que hay en el JSON en forma de un JSONArray.

- WriteFile: crea un nuevo JSON object con los datos de cada corrida al terminar, para así el JSON de data Histórica se actualiza para tener el dashboard de la data importante y comparar cada planta.

- GetDatos: es la clase que lee cada JSON, el de data de la planta al comienzo antes de ejecutarse la interfaz y el de la data histórica al presionar el botón de dashboard en la interfaz.

Conclusión

Viendo los dashboards podemos decir que a pesar de que las dos productoras tienen ganancias similares, los gastos y la producción de capítulos de la productora 1 son mayor, por ende, nuestra mejor productora es la 2. Esto también va a depender significativamente del número de productores de inicio en el caso de la productora 1, y por otro lado, del número de productores de cierre para la productora 2. Además, también se puede observar que en promedio la pérdida del salario del Product Manager de la productora 1 es mayor que el de la productora 2.