A close up of a sign

Description generated with high confidenceInstituto Politécnico Nacional  
Escuela Superior de

Cómputo

Reporte de la práctica 2

Materia:

“Aplicación a Redes”

Profesor:

Rangel Gonzáles Josué

Grupo:

3CV7

Alumnos:

-Escamilla Sanchez Alejandro

-Onofre Resendiz Uriel

Introducción.

Por si quedaba alguna duda, **no son semáforos de tráfico**, es un concepto usado en informática para indicar una estructura con unas características específicas.

Un semáforo es una estructura u objeto que nos va a permitir la sincronización multiproceso o multihilo de nuestro programa. Básicamente se basa en un contador, una función de incrementar el contador y otra de decrementar el contador. El control de acceso se realiza mediante el valor de ese contador, si el contador es negativo el hilo o proceso se suspenderá hasta que ese contador deje de ser negativo, momento en el cual el hilo o proceso empezará a ejecutarse automáticamente.

La implementación de los semáforos varía entre plataformas, por ejemplo, en Linux están implementadas a nivel del sistema operativo dentro de los IPCs.

Si estáis más interesados en cómo funciona un semáforo por dentro podéis leer los problemas: problema del barbero y el problema de los filósofos.

Un semáforo al final funciona de manera similar al letrero del parking que se puede ver en la imagen superior. Mientras que haya plazas libres, deja entrar a los procesos o hilos y cuando hay 0 plazas hay que esperar a que otros salgan para poder entrar.

La función **wait** comprueba que el valor del semáforo, si es negativo o cero el proceso se suspende sin consumir recursos hasta que el valor deje de ser negativo. Si el valor es positivo el proceso o hilo, sigue ejecutándose y se decrementa en una unidad el valor del semáforo. Siguiendo con la analogía del parking, hacer un wait es coger el ticket del aparcamiento cuando hay sitio.

La función **signal** incremental el valor del semáforo, este comportamiento debe realizarse cuando se ha salido de la zona de exclusión mutua. En nuestro caso sería similar a entregar el ticket y salir del parking.

Estas instrucciones pueden modificarse para evitar la espera activa, haciendo que la operación P duerma al mismo proceso que la ejecuta si no puede decrementar el valor, mientras que la operación V despierta a un proceso que no es quien la ejecuta. En un pseudolenguaje más entendible, la operación P suele denominarse "wait" o "espera" y la operación V "signal" o "señal".

El porqué de los nombres de estas funciones, V y P, tiene su origen en el idioma holandés. "Verhogen" significa incrementar y "Proberen" probar, aunque Dijkstra usó la palabra inventada *prolaag*, que es una combinación de *probeer te verlagen* (intentar decrementar). El valor del semáforo es el número de unidades del recurso que están disponibles (si sólo hay un recurso, se utiliza un "semáforo binario" cuyo valor inicial es 1).

La verificación y modificación del valor, así como la posibilidad de irse a dormir (bloquearse) se realiza en conjunto, como una sola e indivisible acción atómica. El sistema operativo garantiza que al iniciar una operación con un semáforo, ningún otro proceso puede tener acceso al semáforo hasta que la operación termine o se bloquee. Esta atomicidad es absolutamente esencial para resolver los problemas de sincronización y evitar condiciones de competencia.

Si hay n recursos, se inicializará el semáforo al número n. Así, cada proceso, al ir solicitando un recurso, verificará que el valor del semáforo sea mayor de 0; si es así es que existen recursos libres, seguidamente acaparará el recurso y decrementará el valor del semáforo.

Cuando el semáforo alcance el valor 0, significará que todos los recursos están siendo utilizados, y los procesos que quieran solicitar un recurso deberán esperar a que el semáforo sea positivo, esto es: alguno de los procesos que están usando los recursos habrá terminado con él e incrementará el semáforo con un signal o V(s).

*Inicia* se utiliza para inicializar el semáforo antes de que se hagan peticiones sobre él, y toma por argumento a un entero. La operación *P* cuando no hay un recurso disponible, detiene la ejecución quedando en espera activa (o durmiendo) hasta que el valor del semáforo sea positivo, en cuyo caso lo reclama inmediatamente decrementándolo. *V* es la operación inversa: hace disponible un recurso después de que el proceso ha terminado de usarlo. Las operaciones *P* y *V* han de ser indivisibles (o atómicas), lo que quiere decir que cada una de las operaciones no debe ser interrumpida en medio de su ejecución.

La operación *V* es denominada a veces *subir* el semáforo (*up*) y la operación *P* se conoce también como *bajar* el semáforo (*down*), y también son llamadas *signal* y *wait* o *soltar* y *tomar*.

Para evitar la espera activa, un semáforo puede tener asociada una cola de procesos (normalmente una cola FIFO). Si un proceso efectúa una operación *P* en un semáforo que tiene valor cero, el proceso es detenido y añadido a la cola del semáforo. Cuando otro proceso incrementa el semáforo mediante la operación *V* y hay procesos en la cola asociada, se extrae uno de ellos (el primero que entró en una cola FIFO) y se reanuda su ejecución.

**Operación atómica**

Una **operación atómica** es una operación en la que un procesador puede simultáneamente leer una ubicación y escribirla en la misma operación del bus. Esto previene que cualquier otro procesador o dispositivo de E/S escriba o lea la memoria hasta que la operación se haya completado.

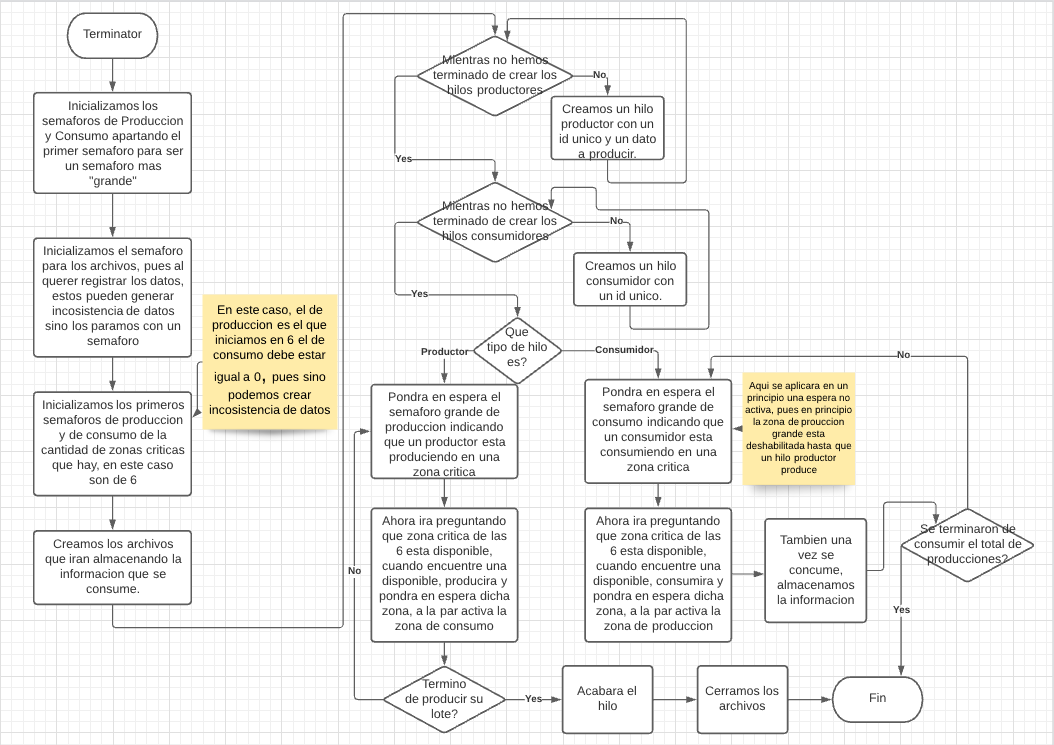
El término atómico implica la indivisibilidad e irreductibilidad del proceso, ya que este debe realizarse en su totalidad o en caso de ser interrumpido poder deshacer sus acciones de modo que fuese como si no se hubiese realizado acción alguna.

# Sincronización

Hablamos de la sincronización de datos como el proceso de alineación entre los datos provenientes de diversas fuentes, y su continua armonización en el tiempo. Vale decir, la coordinación de procesos que se ejecutan simultáneamente, a fin de obtener un orden de ejecución correcto y evitar errores del sistema.

Esta acción permite obtener datos precisos y fidedignos de todos los sistemas, sean estos operativos o transaccionales. Así, cada organización podrá crear una lógica propia para la sincronización de sus datos en tiempo real, si se requiere, obteniendo una mayor precisión, coherencia de la información y una mejora sustancial del rendimiento.

**Diagramas de Flujo:**



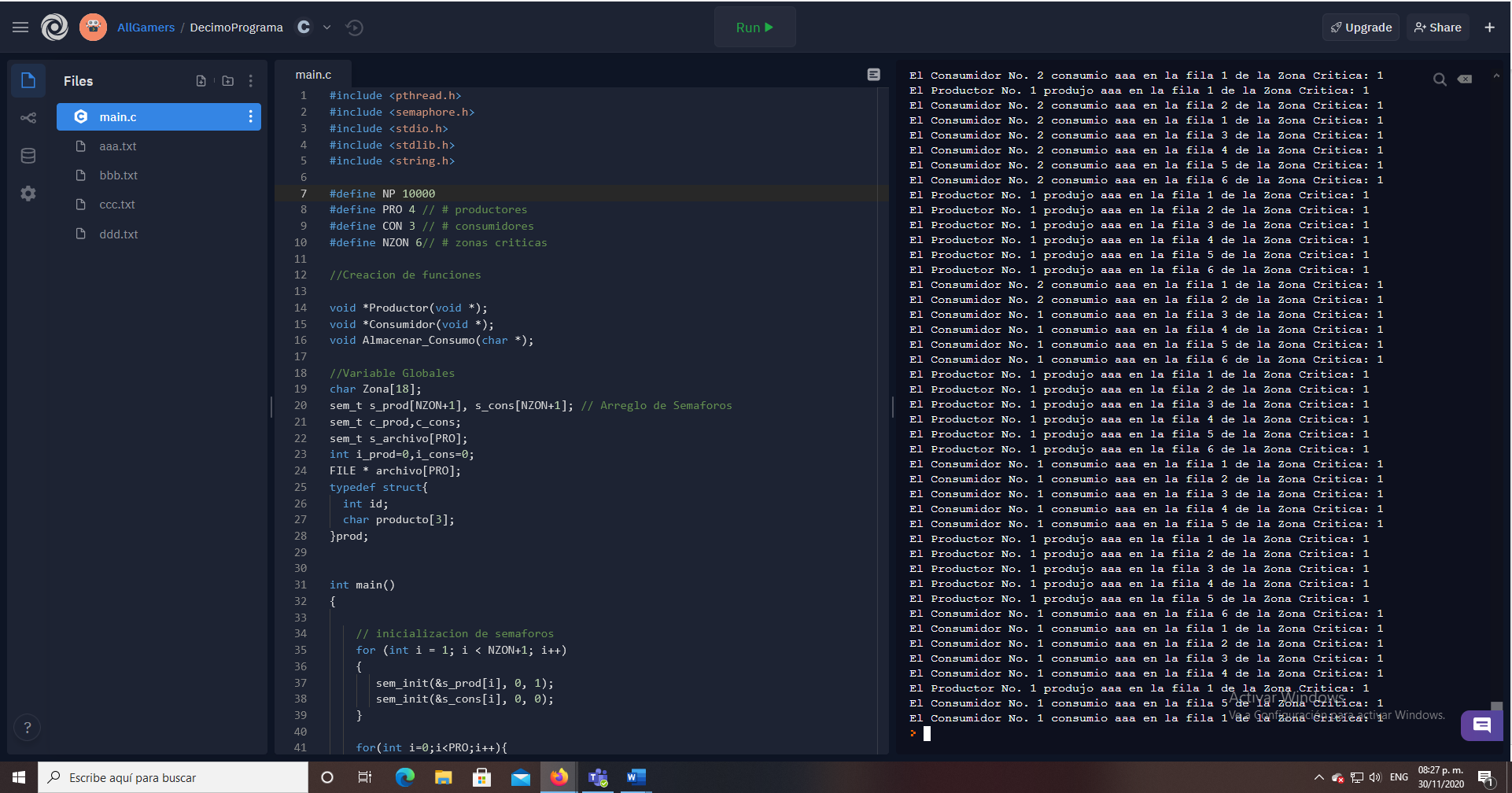
**Explicación:**

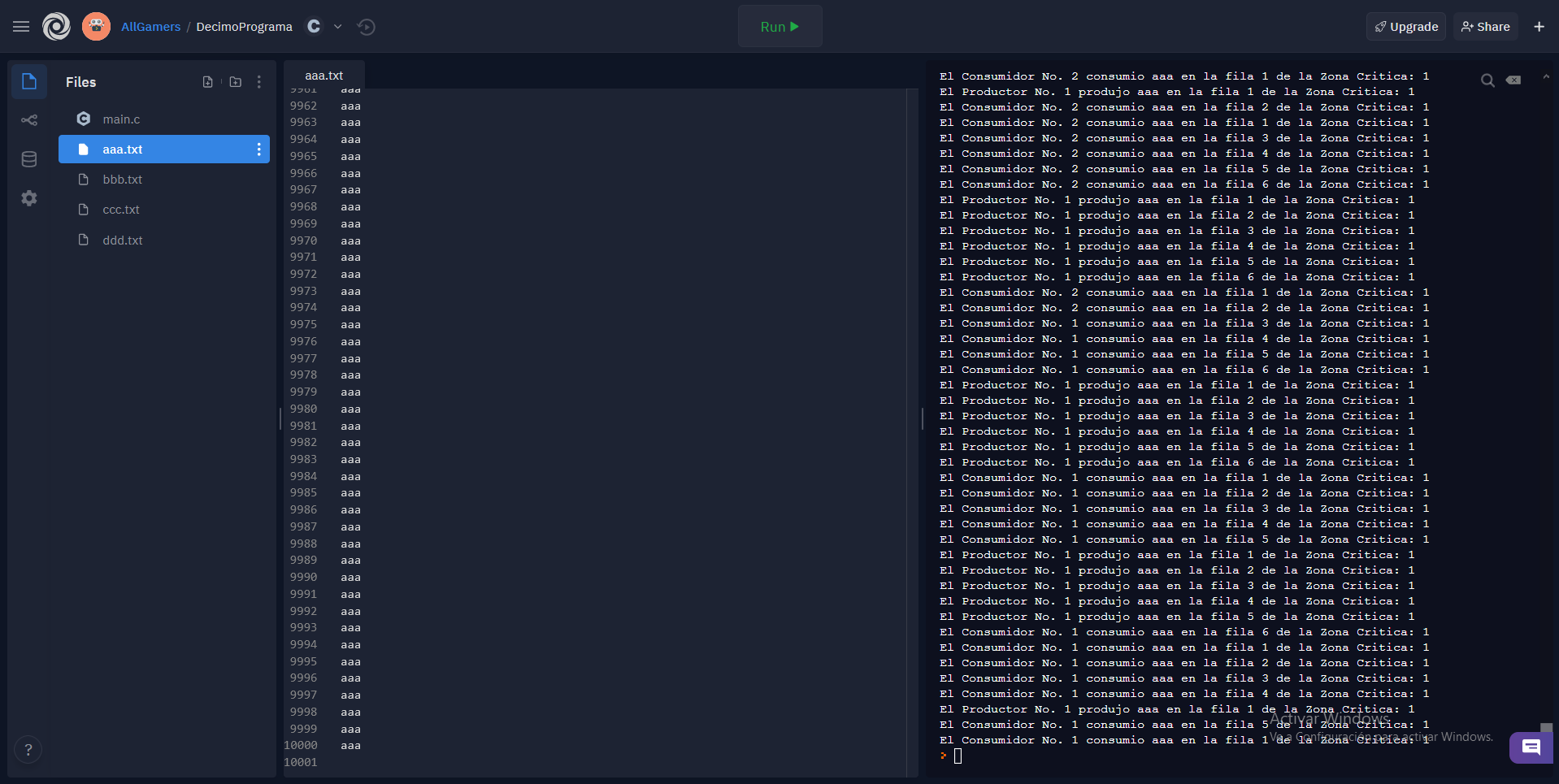
Lo que realizamos es una zona de producción donde tendríamos a 4 productores, 3 consumidores y 6 zonas críticas, para realizar esto hacemos uso de una espera pasiva (a través de la función sem\_wait()) y una espera activa (a través de la función sem\_getvalue()), hacemos uso de varias “ventajas” que nos ofrecen los hilos, propiedades tales como que comparten variables globales entre todos los hilos, haciendo uso de esa ventaja, declaramos varios arreglos globales, tales como:

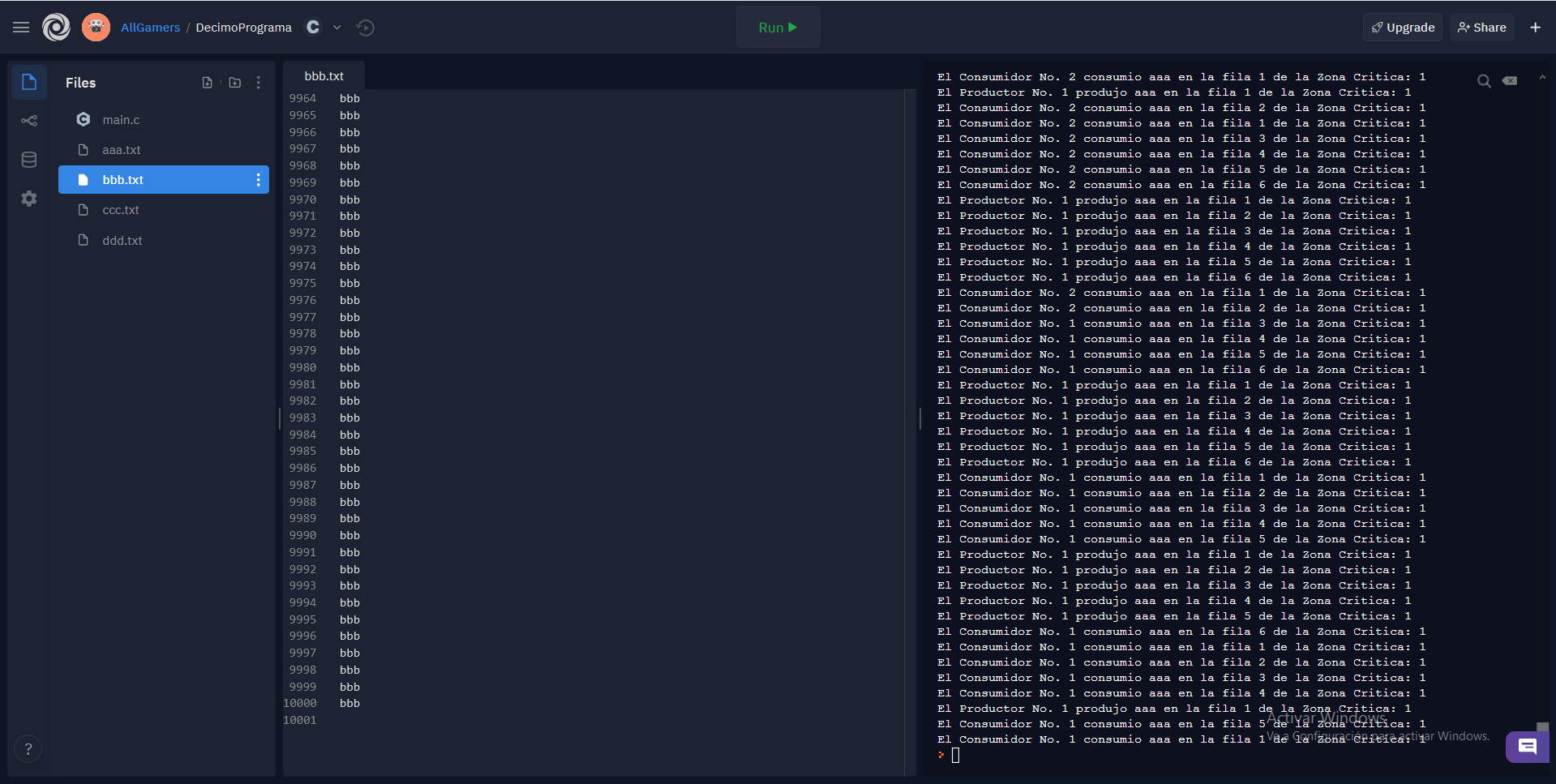
S\_PROD con un tamaño equitativo a el numero de zonas criticas mas una unidad que justificamos mas adelante, realizamos la misma declaración con S\_CONS, realizamos otro arreglo de semáforos que nos servirá para controlar el acceso a los archivos, pues se nos pidió que almacenáramos toda la información que los consumidores consumen, esto con el fin de observar que todo lo que se produce sea consumido y que esto no este ligado de forma que a cada productor le corresponda un consumidor, de hecho es por eso que hay una diferencia entre productores y consumidores para evitar realizar esta solución.

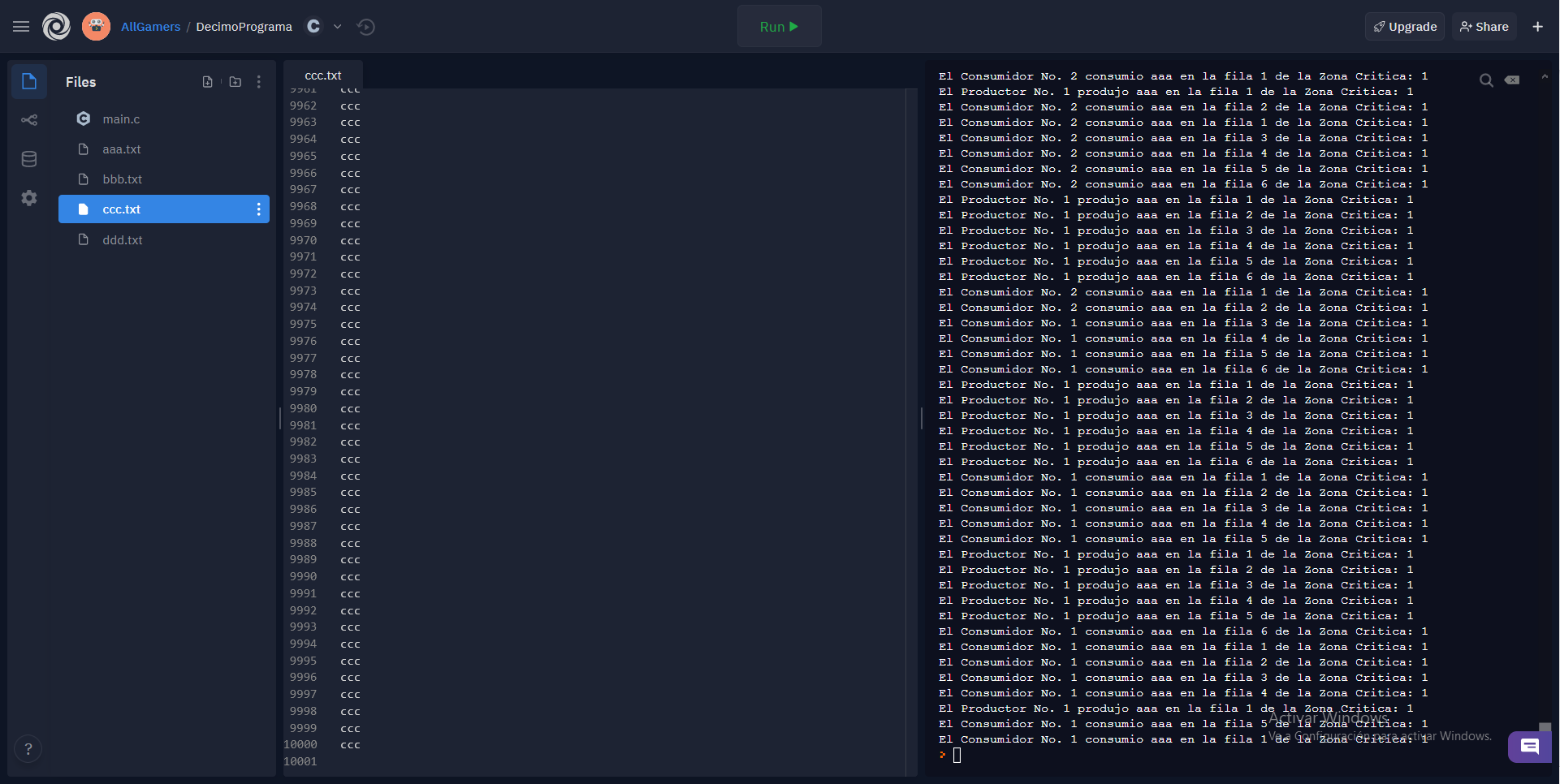
Bueno tomando en cuenta estos arreglos globales de semáforos, comenzamos inicializándolos, tomando en cuenta que S\_PROD y S\_CONS cuentan con una unidad de mas la justificaremos mencionando que para evitar el problema de que todo este “ligado” de cierta forma, hacemos uso de lo que seria un semáforo “grande” que sea del tamaño de la cantidad total de Zonas críticas, esto para limitar la cantidad de productores que entran a producir y que en cierta forma tengan un lugar donde producir, aunque dado que son mas zonas criticas que productores, realmente no se podrán usar las 6 simultáneamente, pero si se podrán usar 4 simultáneamente, esto igual pasa con la producción, todo en producción esta metido dentro de un for, que nos permite llevar el ciclo de tal forma que no saldrá de ese ciclo hasta que efectivamente el productor termine de producir su lote, lo mas interesante reside en la parte de consumidores, pues a falta de que no están ligados con un productor en específico, tenemos que llevar un control para que cuando se llegue al máximo de producción este deje de intentar consumir, esto lo resolvemos haciendo una zona critica mas dentro del lote de consumidores donde llevamos un contador global, el cual una vez llegue a la meta, romperá el ciclo infinito, dado que hacemos uso de uno para mantener siempre a los consumidores en espera hasta que ya no exista producción alguna, de igual forma, aquí mismo es donde igual almacenamos la información que se consume, para esto creamos una función donde por parámetro le pasamos el dato consumido y mediante una comparación localizamos que tipo de dato busca meter en el archivo, dado que pueden llegar varios hilos a dicha función, hacemos uso de los semáforos que declaramos para evitar inconsistencia de datos en casos que aunque pueden ser mínimos, pueden suceder, con esto explicado, solo esperamos a que los hilos lleguen al final y cerramos los archivos a la par que terminamos la ejecución, adelante se mostrara la ejecución con 10,000 ejemplares de producción de cada productor, dándonos un total de 4 archivos con 10,000 datos cada uno.

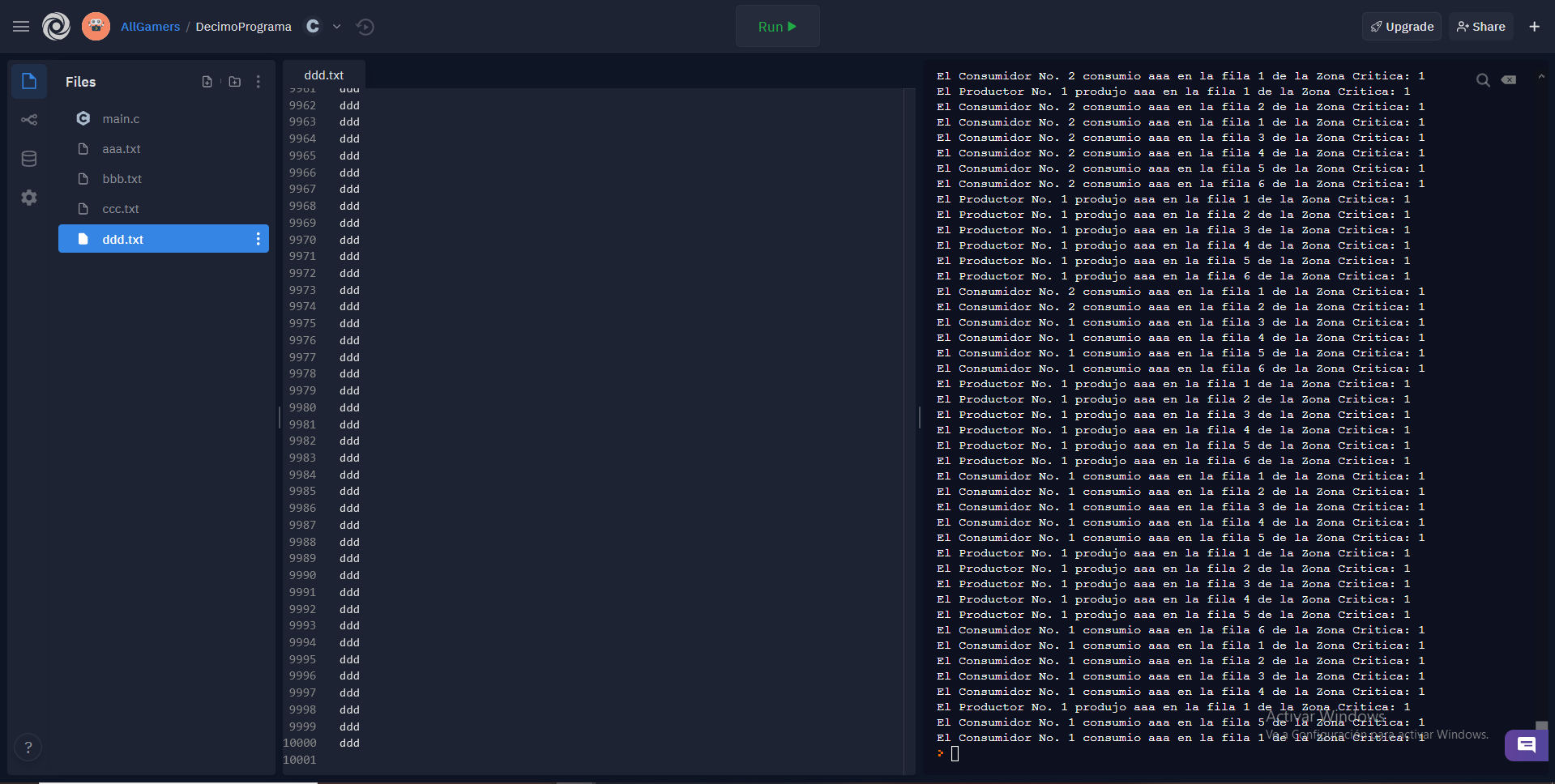
**Ejecución:**











**Conclusiones:**

Onofre Resendiz Uriel:

De la anterior practica podemos concluir que fue bastante laboriosa dadas las especificaciones solicitadas de esta en las cuales se solicitaba una gran cantidad de producciones y un uso adecuado de los semáforos para poder crear una sincronización adecuada, durante esta tuvimos algunas peculiaridades en las cuales casi siempre un productor se quedaba mas apagado de lo habitual y no lograr obtener un lugar. Esta practica a pesar de los ejercicios hechos en clase y las distintas explicaciones para mi se me hizo un tanto difícil pues a mi parecer me hace un poco de falta visualizar el cómo realizar los códigos pero a mi parecer en conjunto a mi compañero se logro obtener un resultado bastante satisfactorio y funcional.

Escamilla Sánchez Alejandro:

Esta práctica, fue la aplicación de todo lo visto teóricamente, nos permitió darle el uso habitual que es “rudo” pues son muchas producciones, sin embargo por el uso adecuado de los semáforos, creamos una sincronización adecuada lo suficiente para que en cualquier caso, este lo pueda sobrellevar de manera adecuada, el detalle es que si suele parecer un “fondo de botella” y mas que nada al final, pues aparentemente casi siempre un productor se queda mas “dormido” de lo habitual y no logra luchar por un lugar hasta que finalmente no hay nadie que pelee mas por las zonas criticas y termina entrando casi de forma secuencial, para mi ese resultado me puede significar dos cosas, la primera es que nuestras propias validaciones realizan un fondo de botella así de grande o la segunda es que siguiendo la teoría a ralla tabla, realmente al momento que realizamos la espera pasiva, este productor se duerme lo suficiente para que cuando se vuelva a iniciar este encuentre el mismo escenario que dejo, por otro lado el consumidor su comportamiento suele ser de acumular “rachas” continuas de consumo, sin embargo mediante el acomodo, logramos hacer que en estas rachas se encuentren otros consumidores, sin embargo no evita del todo las “rachas” y estas siguen ocurriendo.