# Memoria del proyecto de SOPER

Rodrigo Juez Hernández - rodrigo.juezh@estudiante.uam.es
Nicolás Magee Rosado - nicolas.magee@estudiante.uam.es

Mayo 2020

## 1 Diagrama

*Realizar un diagrama que muestre el diseño del sistema, incluyendo los distintos componentes (procesos) y sus jerarquías, así como los mecanismos de comunicación y sincronización entre ellos.*

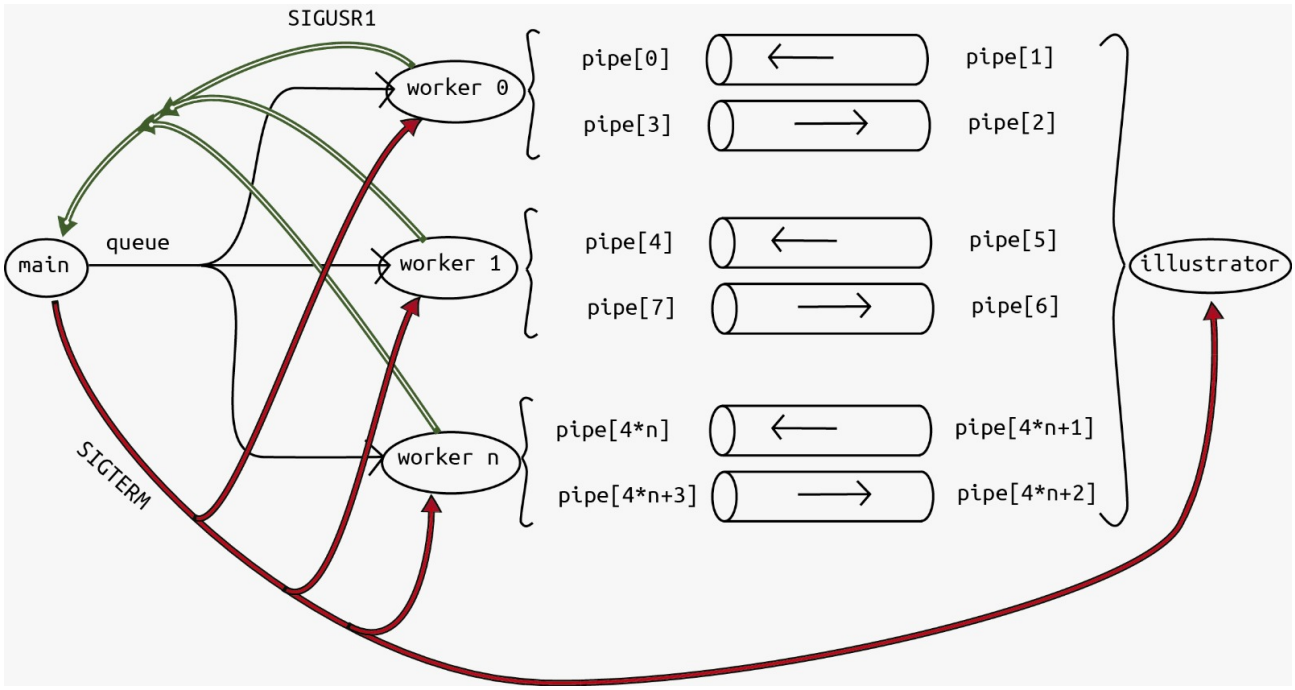
Existen tres tipos de procesos en esta aplicación:

1. Main: el proceso principal "principal" que es el encargado de configurar todo, lanzar, terminar y esperar a los otros procesos que serán sus hijos, y finalmente liberar recursos.
2. Illustrator: el proceso "ilustrador" que se encarga de imprimir por pantalla el progreso del algoritmo de forma periódica a un intervalo fijo marcado por una alarma.
3. Workers: por último hay varios procesos "trabajadores" que son los encargados de ejecutar el algoritmo de ordenación

Para comunicar procesos existen tres mecanismos:

1. Colas: para comunicar el proceso principal con los trabajadores no es posible usar solo tuberías ya que cualquier trabajador puede coger cualquier trabajo de la cola, es decir, cuando el proceso principal manda un trabajo no sabe que trabajador lo va a recibir.
2. Tuberías: en el caso de la comunicación entre el ilustrador y los trabajadores si que se sabe exactamente a quien se quiere mandar mensajes. El uso de tuberías es indicado ya que son más eficientes que las colas y es un diseño más claro.
3. Señales: se utilizan para indicar que se ha terminado un trabajo, por un lado, con SIGUSR1, y para indicar que se debe terminar, con SIGTERMs. El uso de señales está indicado ya que no se pretende comunicar información adicional.

La jerarquía es sencilla. Hay un proceso principal "main" y el resto son hijos suyos.




## 2 Diseño

*Describir el diseño del sistema, incluyendo los aspectos más relevantes, las limitaciones que tenga (si no se han implementado todos los requisitos, es importante especificarlo en este apartado), y los principales problemas que se han encontrado durante la implementación y cómo se han abordado.*

1. Configuración
   (a) Memoria compartida
      i. `shm open`
      Creacíon de región compartida
      ii. `shm unlink`
      Desvinculación: como los procesos que usan esta región son hijos no necesitan acceder por nombre.
      iii. `ftruncate`
      Tamaño para que quepa una estructura tipo Sort.
      iv. `mmap`
      Asociar un rango de direcciones a la memoria compartida para acceder a ella.
      v. `init sort`
      Inicializar la estructura con la función provista.
   (b) Handlers `sigaction`
   Se configuran cuatro manejadores de señales para cada señal utilizada: SIGTERM, SIGUSR1, SIGINT, SIGALARM
   (c) Cola
      i. `mq open`
      Creacíon de la cola.
      ii. `mq unlink`
      Desvinculación: No se va a referir por nombre (al igual que la zona compartida).
   (d) Tuberías `pipe`
   El número de tuberías es el doble del número de trabajadores.

(e) Mutex `sem init`
Un semáforo en memoria compartida que empieza con valor a 1.

2. Ilustrador
El ilustrador, mientras no reciba una señal SIGTERM, ejecuta un bucle:

   Primero lee de cada cola, guardando lo que lee en un array y una vez leídos todas las tuberías representa el valor por pantalla.

   Una vez impreso, manda (también por tuberías) una señal para que puedan continuar los trabajadores.

   Cuando recibe una señal para terminar, lo hace de forma controlada cerrando los pipes y liberando recursos.

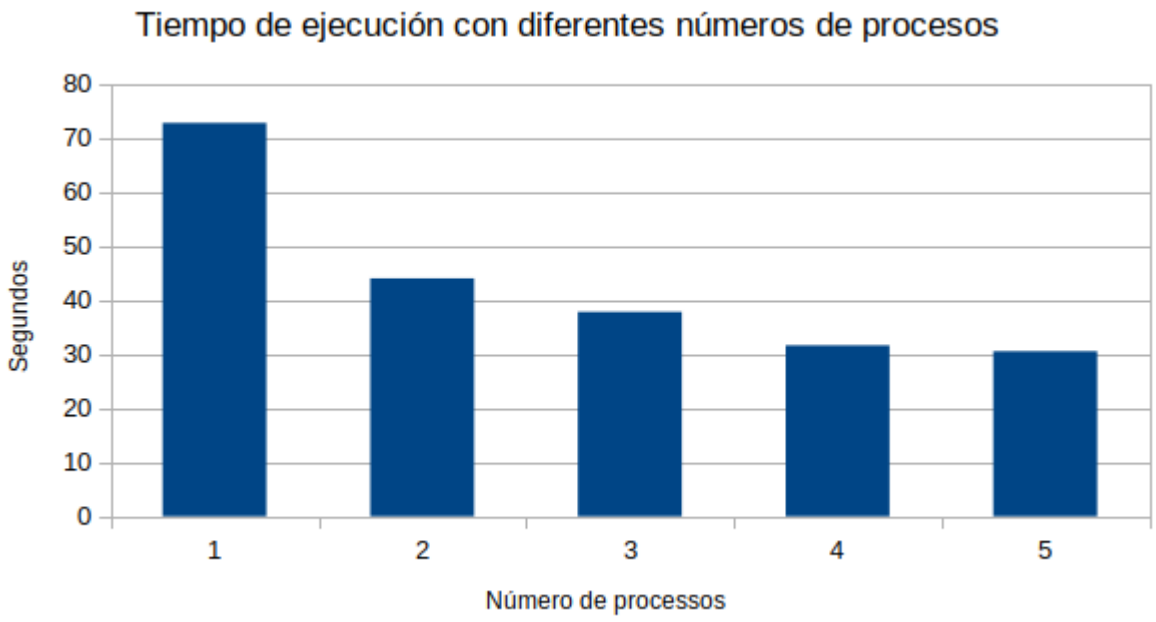
3. Trabajador
Los trabajadores tienen que leer de la cola para recibir y asignarse un trabajo. Al leer tiene que comprobar si sale de la espera por recibir una señal.

   A la hora de acceder a partes del struct compartido de Sort, fuera de la franja de los datos adjudicada, utiliza el mutex.

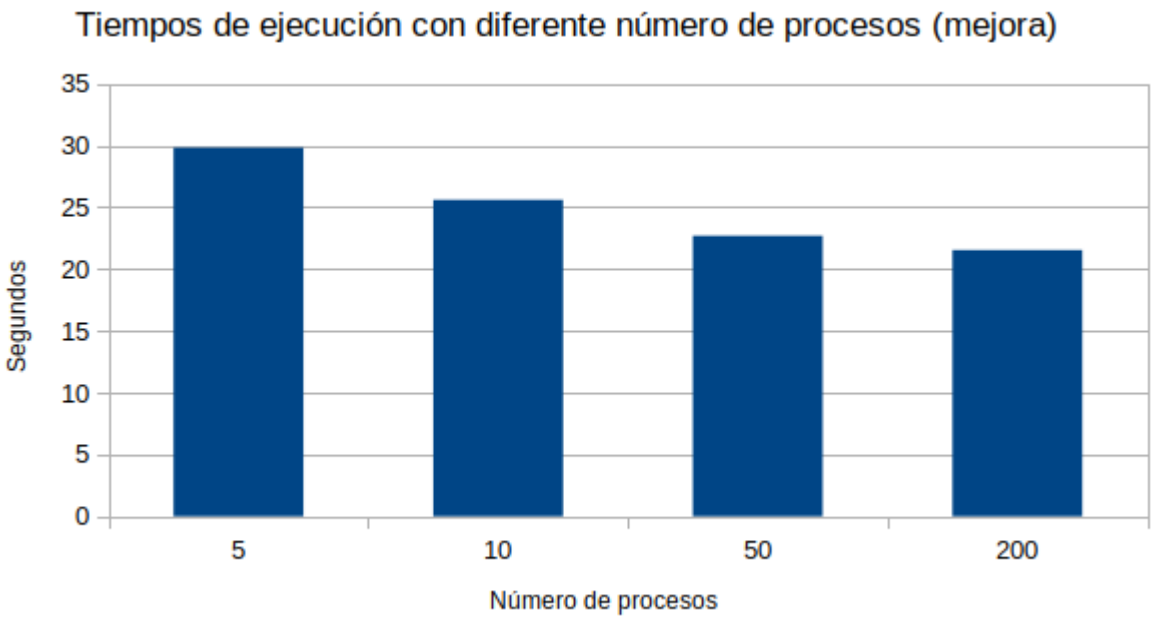
# 3 Tiempos

*(Opcional) Comparar los tiempos de ejecución del sistema multiproceso con distintos números de procesos. Para ello, fijar el número de niveles a 10 y pintar una gráfica que muestre el tiempo de ejecución respecto al número de procesos. Comentar la evolución que se ve en esta gráfica.*

Se puede observar que cada vez que se añade un proceso el tiempo de mejora va disminuyendo. Se ha probado usando DataMedium con delay de 15.




Tiempos después de hacer la mejora opcional y poder realizar trabajos sin haber terminado por completo el nivel actual.




Se puede observar una mejora, puesto que antes, pasados los 4 procesos, apenas veíamos una mejora, ahora se observa que hay una mejoría desde 5 hasta 50, pasado de ahí, debido a que tenemos pocos niveles y las tareas no se fragmentan tanto, no obtenemos mucha mejoría.