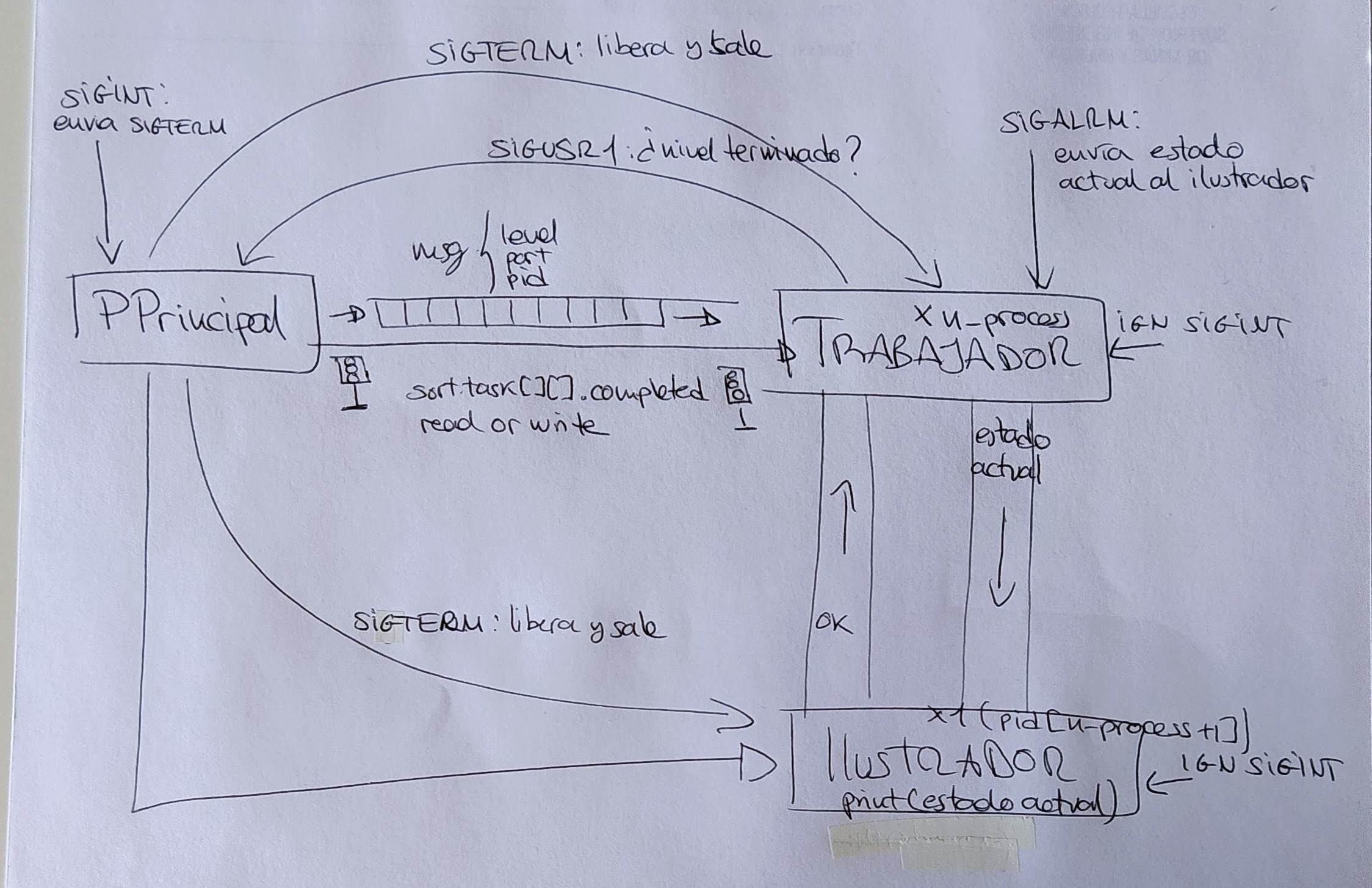
**PROYECTO**

**a) 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.**

****

**b) 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.**

El sistema consta de un proceso principal, que crea n\_process trabajadores y un ilustrador.

El proceso principal manda a través de una cola de mensajes el nivel y la parte que tiene que ordenar cada trabajador. Así mismo, en una memoria compartida, va poniendo si una tarea ha sido completada o no. Esto es controlado mediante semáforos.

Si el proceso principal recibe la señal de SIGINT (tanto los hijos como el ilustrador ignoran esta señal) envía la señal de SIGTERM a el proceso ilustrador y a todos los trabajadores, y liberan los recursos apropiadamente y terminan.

Cada trabajador se bloquea a la espera de una tarea que tiene que completar a través de la cola de mensajes. Esta tarea, que es enviada por el proceso principal, al terminar el trabajador pone como completada en el segmento de memoria compartida y se envía al padre la señal de SIGUSR1. EL padre comprueba que se han terminado todas las tareas de ese nivel, si es así, se envían las tareas del siguiente nivel.

El trabajador ignora la señal SIGINT, pero cuando el proceso principal le envía la señal de SIGTERM libera y termina su acción.

Si recibe la señal de SIGALRM (cada segundo) envía su estado actual al ilustrador a través de una tubería y se bloquea a la espera de la respuesta del ilustrador.

El ilustrador recibe el estado de los trabajadores y los imprime por pantalla, al igual que el estado del programa. También ignora la señal de SIGINT. Cuando haya impreso el estado, mandará por otra tubería al trabajador, que se ha recibido la información correctamente y que este puede seguir su trabajo.

Uno de los problemas encontrados fue a la hora de manejar una señal. Para algunas hemos incluido una flag y hemos hecho diversas comprobaciones en la función principal para controlarlo. Para otras hemos creado variables globales y hemos realizado la acción correspondiente dentro del manejador.

Otro de los problemas fue la comprobación de SIGUSR1 en el proceso principal para comprobar si las tareas de ese nivel habían sido realizadas. Para ello decidimos bloquear la señal mientras el proceso principal quedaba bloqueado en mq\_send para que no hubiera interrupciones ya que si queda bloqueado es porque aún quedaban tareas por realizar en ese nivel. Y posteriormente un for que comprobase el estado de todas la tareas del nivel, si alguna no estaba completa se queda bloqueado a la espera de una señal SIGUSR1 o SIGINT y vuelve a comprobarlo si le llega una SIGUSR1.

El mayor problema y el que no hemos conseguido resolver han sido las tuberías. Tuvimos éxito al crear un proceso extra para el ilustrador pero no para enviar correctamente ni recibir los mensajes. Los mensajes de error fueron “broken pipe”, “bad file descriptor” y “interrupted system call” (estos errores nos dieron en distintas partes ya que íbamos probando nuevo código). Por eso existen dos carpetas: una funcional que completa hasta el punto 6 de implementación y otra con el intento que hemos hecho para implementar las tuberías y el ilustrador. De esta última, en mi opinión es que el planteamiento está bien (creando el proceso extra y creando una nueva alarma cada segundo una vez se ha recibido el mensaje del ilustrador en el manejador de SIGALARM) pero fallamos en alguna parte de las pipes, ya sea la creación, la lectura o la escritura.