TP6 PSCD

2017-2018

DISEÑO DE LA APLICACIÓN

García Hernández, Alberto 741363
Generelo Gimeno, Jorge 737317
Gómez Lahera, Miguel 741302

INDICE:

1. Diseño interno de los procesos del sistema:

- En cuanto al diseño interno del sistema se ha organizado en tres directorios: uno para el cliente, otro para el servidor y un último para las librerías utilizadas en la implementación del sistema.

- En cuanto al directorio de las librerías encontramos las librerías suministradas para la gestión de las imágenes a mostrar en las vallas y las librerías de sockets y semáforos proporcionadas para la realización de las prácticas de la asignatura.

- El directorio del cliente contiene los ficheros encargados de ejecutar tanto el cliente manual como el cliente automático. El cliente se basa en el envío de mensajes al servidor solicitando una operación. Así pues, en cuanto el cliente establece conexión con el servidor se le pregunta si desea participar en la subasta, a lo que el cliente deberá responder “SI” en caso afirmativo, o “NO” en el caso contrario. Una vez dentro de la subasta el cliente se le informa sobre el estado de la subasta en ese momento, y desde ese instante tiene ya la posibilidad de pujar, solicitar información acerca de la subasta o pedir ayuda sobre el funcionamiento de esta.

- En el directorio del servidor encontramos los módulos que se encargan de gestionar las subastas y las vallas. Así pues, tenemos: el módulo Administrador encargado de controlar toda la información relacionada con la subasta (modulo subasta y gestor), el módulo Gestor cuya función es obtener los enlaces de las imágenes a mostrar y descargar las imágenes, y el módulo Subasta, que lleva a cabo la organización de la subasta y un estadístico sobre el servidor desde el momento de su inicio. Por otro lado, se ha utilizado un monitor para garantizar la exclusión mutua a la hora de gestionar los datos del módulo subasta con la intención de evitar problemas al llevar la cuenta de las distintas pujas (tales como acceso indebido a variables, sobrescritura de datos, inanición, … etc.). Por último se ha definido la clase valla para facilitar el manejo y la presentación de las vallas.

- Además, tanto en el servidor, como en el cliente, están protegidos ante la señal “Ctrl + c” (SIGINT en sistemas linux), de forma que ante este comando, ambos, siguen funcionando.

1.1. Cliente:

- Antes de responder a la pregunta de si desea participar en la subasta o no, trata de conectarse al servidor, en caso de hacerlo, crea y lanza un proceso “esc”. Este se encarga de recibir los mensajes provenientes del servidor, así como de enviar las sucesivas órdenes que el cliente quiera ejecutar. Este proceso se mantiene en curso hasta que o bien el servidor se cierre o el cliente quiera salir.

- La URL de la imagen debe ser introducida como parte de los parámetros de entrada del script del cliente manual.

- El cliente dispone de una función para validar si el mensaje introducido por el cliente es válido, en el caso en el que no lo sea se le avisara a el cliente, y ademas se le permitirá volver a introducir el mensaje. En el caso en el que se introduzca un mensaje inválido y la función no sea capaz de reconocerlo dependerá del servidor de informar al cliente de lo sucedido.

1.2. Servidor:

- El servidor comienza creando un administrador, con sus respectivos gestor y subasta. A continuación crea los siguientes cuatro primeros procesos: administrador, gestorP, gestorP2 y subastador.

- El proceso administrador llama a la función “iniciarAdmin” del módulo administrador, que consiste en un bucle que lee constantemente cualquier posible orden ejecutada por la línea de comandos del servidor (HISTORICO O ESTADO) hasta que el usuario escriba la orden EXIT. En este último caso

procederá a cerrar el servidor de forma ordenada, impidiendo que se puedan conectar nuevos clientes al servidor y además, forzando el cierre del módulo Subastador, que esperará a que finalice la subasta actual, pero se cerrará antes de que tenga lugar una nueva ronda. Una vez que la última subasta haya tenido lugar se notificará a los clientes de que el servidor se va a cerrar, enviándoles en mensaje de finalización, y que por lo tanto ya no van a ser atendidos. Una vez que todos os clientes se hayan desconectado se procederá a cerrar el servidor, siempre y cuando no falten vallas por mostrar, en cuyo caso se esperará a que el gestor termine de mostrar todas las vallas restantes, y una vez que este haya finalizado su ejecución el servidor se podrá cerrar.

- El proceso gestorP y gestorP2, hacen referencia a las dos vallas publicitarias disponibles. La implementación de esta está dentro de Gestor.cpp (función Gestor :: iniciar). Estos dos procesos son los encargados de gestionar los accesos a la cola de vallas mediante semáforos y a su vez, van mostrando las que haya disponibles en la cola durante el tiempo especificado (se ira mostrando cada imagen en una posición distinta de la pantalla). Para evitar las esperas activas estos dos procesos se pondrán en espera mientras no hay ninguna valla que mostrar. En el momento que la variable “fin” está a true, rechaza nuevas vallas y espera a que terminen para luego poder cerrar el servidor.

- El proceso subastador llama a la función gestorSubasta. Como su nombre indica, esta es la encargada de gestionar el correcto transcurso de las sucesivas subastas que tendrán lugar. Su diseño se basa en un bucle que acaba en el momento que la variable “fin” tome valor true, a causa de que se haya introducido la orden EXIT, desde el módulo del administrador. Comienza su ejecución creando una subasta e inicializándola con unos valores generados aleatoriamente. Entonces esperará a que todos los clientes pasen su turno para poder dar la subasta como finalizada. En ese momento el procedimiento “cerrarSubasta” se desbloqueara, y en el caso de tener un ganador, lo anunciará. Además parará la ejecución del resto de

procesos, mientras el cliente que ha resultado ganador le envía la URL de la imagen que quiere mostrar en la valla. Toda la información respectiva a la valla publicitaria (URL, path y duración) se ha encapsula en el tipo de dato Valla. Una vez finalizada una subasta se reinician sus valores y se repite la ejecución.

- Después de haber lanzado estos procesos, el servidor comienza a aceptar clientes, y en función de los que entran lanza el proceso que llama a la función “servCliente”. Para lanzar un número indeterminado de estos nuevos threads encargados de atender a nuevos clientes, hacemos uso de la función detach(), de la librería estándar de threads, que nos permite desasociar el thread (objeto) de la ejecución de la función, pudiendo reutilizarse indefinidamente, aunque listen(max_connections) sigue limitando hasta un cierto punto esta funcionalidad. Esto puede causar problemas de sincronización, al no poder hacer un join() al final de la ejecución de la main para que espere a que finalicen todos los procesos y es por ello que el monitor se deberá encargar también de esta misma tarea, la cual se gestiona a través de las llamadas a Entrar, Salir y Finalizar, las cuales impiden que el proceso principal avance hasta que todos los clientes hayan finalizado.
- Aquí son introducidos los nuevos clientes donde serán atendidos por un proceso encargado de recibir los mensajes provenientes de cada cliente, es decir, sus órdenes, e interpretarlos para su correspondiente respuesta o ejecución.

2. Protocolos de interacción que han sido implementados:

- En cuanto a la logística del sistema, los mensajes intercambiados entre servidor y cliente se basan en una palabra en letras mayúsculas indicando la operación que se desea realizar. Al cliente, al establecer conexión con el servidor, se le pregunta si desea participar o no en la subasta. Una vez acceda a la subasta (en caso de que así lo indique), el servidor le envía la información de la subasta que está abierta en ese momento. El cliente puede realizar su puja escribiendo por la línea de comandos “PUJAR” seguido del valor que quiere poner en la subasta siempre y cuando sea superior a la puja mínima al lanzar la subasta o la última puja realizada. En caso contrario la puja no será aceptada y se le muestra de nuevo la información de la subasta en ese momento. En el caso de que un cliente no quiera pujar en la ronda de la subasta actual tiene que escribir el mensaje “PASO”, y no participará en la ronda actual. La subasta en curso solo acabará en el momento en el que todos los clientes pasen, hayan realizado apuesta o no. Al finalizar la subasta si el cliente con la mayor puja no ha superado el mínimo oculto de la subasta, no será considerado ganador. Si su puja sí que supera ese mínimo, el servidor le pedirá la URL de la imagen que quiera mostrar en la valla publicitaria contratada, y este se le enviará para que pueda descargarla y mostrarla durante el tiempo acordado. Basándonos en cómo está implementado, el URL de la imagen debe ser introducido correctamente es decir, corresponder a un enlace a una imagen que pueda ser descargada, en el caso contrario se producirá una excepción en la librería CImg.

- Otros mensajes que puede seleccionar el cliente para enviar al servidor son “ESTADO”, “AYUDA” o “EXIT” que se encargan de pedir el estado de la subasta actual al servidor, mostrar todos los comandos posibles a enviar o cerrar la conexión con el servidor, respectivamente. El servidor también cuenta con varias opciones para obtener información sobre el sistema, los cuales son: “HISTORICO” Y “ESTADO” que muestran la estadística histórica del sistema y el estado de la subasta actual, respectivamente. Esta información es obtenida de los módulos Subasta y Monitor.

3. Mecanismos de gestión de concurrencia:

- Para conseguir sincronizar correctamente todos los procesos y gestionar los problemas de concurrencia asociados, se han utilizado varios recursos de los vistos en clase. El módulo de monitor junto a varios semáforos serán los encargados de garantizar que se cumplan las propiedades de exclusión mutua y evitar cualquier posible problema en la ejecución del programa, debido a la inanición de alguno de sus procesos. Todos estos recursos están basándonos en la librería de semáforos proporcionada en clase y en las bibliotecas de condition variable y mutex.
- El monitor es usado para garantizar la exclusión mutua a la hora de acceder a las variables compartidas de la subasta, para acceder a estas se emplearán operaciones implementadas en el monitor como pujar, evitando que al realizar las pujas algunos valores y mensajes se superpongan unas con otros, se lleguen a sobrescribir o algún proceso sufra un bloqueo o inacción. Junto a la función de pujar el monitor dispone de distintas operaciones como Entrar, Salir y Finalizar que gestionan las entradas y salidas de los clientes y que garantizan que el servidor no finalizará hasta que todos los clientes se hayan desconectado.
- A su vez también dispone de operaciones como “siguientePuja”, “bloquearSubasta” y “desbloquearSubasta”, que son utilizadas para gestionar el transcurso de cada subasta, garantizando que solo se pasará de ronda cuando todos los clientes han enviado un mensaje, y que a su vez, la subasta solo finalizara en el caso en el que todos los participantes hayan decidido pasar en la misma ronda.
- También se recurre a los semáforos, que se usan en varias partes de la implementación con la finalidad de resolver problemas de concurrencia más simples. En el módulo gestor se utilizan tres semáforos, uno para asegurar la sección crítica al acceder a la cola de vallas que contiene este módulo, otro para esperar cuando no haya ninguna foto que mostrar en las vallas, y por último, otro para garantizar que no se cerrara el gestor hasta que que ambos procesos hayan finalizado.