Diseño de una aplicación de chat entre dos computadores.

Además de las funcionalidades imprescindibles que debía de tener la aplicación se han implementado o tratado con profundidad algunas otras.

Los comentarios añadidos al código para facilitar la lectura están sólo en el programa cliente (ya que el código es exacto casi en su totalidad con el del servidor y así no fue necesario repetirlos).

Características implementadas:

- El código está estructurado en tres hilos más el main: flujoOut es el encargado de enviar mensajes, flujoIn de recibirlos y ficheroOut de enviar los ficheros. Los dos primeros se declaran y se arrancan desde el main y el de ficheroOut se arranca desde el hilo de flujoOut (cuando se indica el envío de un archivo). Para recibir un archivo que el otro usuario me envía uso el hilo flujoIn (donde distingo entre si el texto recibido contiene la cabecera de archivo o si es texto llano que debo mostrar por pantalla). Todos los hilos usan un mismo socket.
- Hay dos versiones de la aplicación (cliente y servidor), sin embargo el código es exactamente el mismo para ambas salvo en la parte del main donde el socket se crea de forma distinta para el cliente que para el servidor. Al arrancar la aplicación cliente se solicitará al usuario que indique la IP del otro usuario (IP destino); podrá usarse la 127.0.0.1 (localhost) para realizar pruebas en el mismo ordenador. El puerto destino no se solicita ya que se ha especificado uno manualmente, concretamente el 8081 (más o menos arbitrario y modificable en cualquier momento).
- Una vez se ha arrancado la aplicación (primero el servidor y luego el cliente) el cliente deberá mandar el primer mensaje ya que el servidor está a la espera de recibir una conexión por parte de un cliente en el puerto indicado.
- Tanto a servidor como a cliente se les indica que han iniciado un chat con otro usuario, que para iniciar una transferencia de archivo deben escribir el comando @rchivo y que para terminar deben escribir fin.
- El proceso de cierre de la aplicación consta de una negociación no negociable (el otro no puede negarse) en la que la parte que solicita el cierre le indica a la otra que ya no va a enviarle más mensajes (el solicitante envía finrequest y cierra flujoOut) y que por tanto el otro puede cerrar su flujo de recepción de texto (flujoIn), a continuación la otra parte se asegura que ya no va a mandarle nada más y tras estar seguro de ello se lo indica al solicitante (le envía finaccepted al solicitante y cierra flujoOut) para que éste pueda cerrar también su flujo de recepción de texto.
  Cabe la posibilidad de que mientras hubiera una transferencia en curso se solicitara el cierre por cualquiera de las dos partes por lo que a la parte solicitante del cierre se le preguntará si realmente quiere terminar dicha transferencia (el usuario deberá contestar SI o NO y se le seguirá preguntando mientras no introduzca una de estas dos opciones). En caso de que quiera interrumpir la transferencia se llevará a cabo un

proceso de cierre organizado de la transferencia para posteriormente proceder a la negociación de cierre habitual.

- Cuando una de las dos partes introduzca el comando @rchivo se estará indicando el inicio de una transferencia y al solicitante se le preguntará el nombre del archivo origen (deberá escribir elnombre.txt) y en caso de que no pueda encontrarlo se lo indicará al usuario, éste podrá intentarlo nuevamente volviendo a introducir el comando @rchivo. Si el usuario hubiese introducido el comando @rchivo sin querer o se lo hubiera repensado podría cancelar el proceso escribiendo CANCELAR. Si ha podido encontrar el archivo se le indicará al otro usuario que le quieren enviar un archivo cuyo nombre en origen es elnombre.txt, éste podrá escoger entre guardarlo como ciertonombre.txt o CANCELAR para rechazarlo. En caso de que lo haya rechazado se le indicará al solicitante que le han rechazado la oferta y la conversación seguirá normalmente (pudiéndose realizar otra petición de archivo acto seguido si se desea). En caso de que el usuario quiera aceptar el archivo deberá indicar un nombre, luego se le preguntará si es correcto (donde deberá escribir OK o indicar otro nombre tantas veces como quiera) y acto seguido se indicará tanto al emisor como al receptor que se ha iniciado la transferencia con un mensaje del tipo “Transferencia iniciada”. Ambas partes pueden seguir conversando mientras el archivo se transfiere. Cuando la transferencia haya terminado se les indicará a ambas partes con un mensaje del tipo “Transferencia terminada”.
- Si se indica el comando @rchivo por cualquiera de las dos partes cuando hay una transferencia en curso saldrá un mensaje indicando que deberá esperarse hasta que termine la transferencia actual para iniciar otra de nueva (una futura implementación sería gestionar el envío de múltiples archivos a la vez).
- Las respuestas del usuario son sensibles a mayúsculas por lo que deberá atenerse a lo que le pidan que conteste. Si le piden que indique SI o NO deberá indicar eso y no servirá que indique “Si” o “si” o cualquier otra combinación ya que la máquina no dará la respuesta como válida y se le seguirá preguntando hasta que conteste correctamente una de las dos opciones. Lo mismo pasa con CANCELAR (no sirve que indique cancelar en minúsculas).

Para realizar las pruebas de transferencia de archivos he creado un fichero de texto de 400MB para que me diera suficiente tiempo como para comprobar las distintas funcionalidades implementadas. Normalmente los ficheros de texto no ocupan tanto y la transferencia es muy rápida, sin embargo este programa se ha realizado pensando en que en vez de el envío de texto podría hacerse el envío de ficheros comprimidos o ejecutables, así como imágenes o documentos PDF de más peso.