75:42 - Taller de Programación I

Ejercicio Nº ___ Padrón _____

Alumno _____ Firma _____

Nota:		**Corrige**:		**Entrega #1**
				Fecha de entrega
				Fecha de devolución

Nota:		**Corrige**:		**Entrega #2**
				Fecha de entrega
				Fecha de devolución

El presente trabajo, así como la entrega electrónica correspondiente al mismo, constituyen una obra de creación completamente personal, no habiendo sido inspirada ni siendo copia completa o parcial de ninguna fuente pública, privada, de otra persona o naturaleza.

Índice general

1 Changelog

1.1. Correcciones pedidas

1. No está bueno que haya uso de sockets en los main del cliente y servidor, debe tratarse dentro del protocolo.
2. En SocketConnect no se utilizan excepciones.
3. La forma de parsear comandos no escala, podría haber una entidad que se encargue de este proceso tanto en el cliente como en el servidor. Utilizar polimorfismo.
4. Hardcoding de cantidad y posición de argumentos, y códigos de protocolo.
5. Crear el certificado desde un archivo en common_certificado podría hacerse de manera más compacta, sin duplicación de código.
6. Investigar las opciones de la biblioteca ctime para el manejo de las fechas.
7. Uso innecesario de punteros en los atributos de SocketAccept.
8. Como bien notaste: Las clases Claves de server y client son idénticas.
9. Falta la sobrecarga de los operadores « y ».
10. Falta verificar los clientes que terminaron su ejecución para eliminarlos del vector de punteros a Comunicadores y que no crezca indefinidamente. Por esto tener un vector como contenedor no está bueno ya que al momento de eliminar uno que haya terminado su trabajo, el vector realoca todos los comunicadores siguientes al eliminado a su nueva posición, lo cual es ineficiente.
11. No llamar explícitamente al destructor de SocketAccept.

1.2. Correcciones hechas

- Se lanzar errores en caso de que se pierda la conexión durante un envió, durante una recepción de un mensaje y en caso que no pueda realizarse una conexión satisfactoria. El mismo se atrapa en el caso del cliente en el main ya que no hay mucho que se pueda hacer, y en caso del servidor en la clase Comunicador, de forma que el server pueda seguir recibiendo otros clientes. [2]

- Se cambiaron las funciones del protocolo de forma que se envíen y reciban cadenas de un tamaño máximo.[4]
- Para poder procesar el certificado se pudo extraer la lógica que tenían en común cada función que identifica un campo. [5]
- Se pudo utilizar la biblioteca ctime y remover el uso innecesario de un mapa con los nombres de los meses. [6]
- Se pudo cambiar el uso de punteros para los sockets de conexion, se definio un constructor por movimiento y se elimino el constructor por copia, esto resulto muy util tambien para la clase AceptadorDeClientes ya que la clase socketAccept devuelve un nuevo socketConnect y con las operaciones definidas el compilador automaticamente sabe que se debe realizar un move cuando se llama a la funcion acceptSocket, no hizo falta el uso de punteros y tambien se ahorro el trabajo de hacer new y delete evitando una mayor posibilidad de obtener leaks. [7]
- Se juntaron las clases Claves de server y client en una clase comun. [8]
- Se pudieron sobrecargar los operadores pedidos [9]
- Se utilizo una lista enlazada (la lista estandar de c++) para poder ir removiendo los threads que terminaron, de esta forma no hay problemas como ocurria en el caso de un vector cuando se trataban de realocar los demas threads cuando uno se removia de la lista. [10]

1.3. Correcciones faltantes

En ambas ocaciones el problema fue la falta de tiempo, para proximos trabajos trataria de plantear mejor el diseño para no tener que realizar tantos refactors. Estas correcciones son:

- No está bueno que haya uso de sockets en los main del cliente y servidor, debe tratarse dentro del protocolo. [1]
- La forma de parsear comandos no escala, podría haber una entidad que se encargue de este proceso tanto en el cliente como en el servidor. Utilizar polimorfismo. [3]