Trabajo Práctico: Fase 1 - Comunicaciones

Objetivo

Desarrollar una aplicación Cliente-Servidor constituida por un programa llamado Servidor que recibe, procesa y responde mensajes a uno o más programas llamados clientes.

El Servidor

Se deberá desarrollar un programa en C/C++ que lea desde un archivo en formato XML la definición de un puerto y la cantidad de clientes que pueden ser atendidos simultáneamente. Si el archivo no existe, se creará uno por defecto. El nombre del archivo será pasado por línea de comandos.

Una vez parseado el archivo XML, el servidor deberá gestionar la mensajería de una sala de chat y buzón de mensajes. En caso de superar el máximo de clientes se deberá informar al programa cliente. Los datos de autenticación de los usuarios están en el archivo de configuración xml.

Además, el servidor deberá almacenar en un archivo de texto plano información pertinente sobre la actividad que realiza, así como los errores encontrados, problemas de comunicación, etc. Se deberán loguear distintos tipos de mensajes según alguno de los 3 niveles de log establecidos:

DEBUG: Información detallada de todo lo que realiza el servidor.

ACTIVIDAD: Muestra poca información sobre la actividad que realiza el servidor.

ERROR: Muestra información sobre los errores que se producen en la ejecución.

El servidor ejecutándose por consola, deberá presentar un prompt para que el usuario pueda ejecutar comandos. Los comandos posibles son:

1) Close-server: cierra todas las conexiones con los clientes y libera los recursos de manera correcta. En caso de tener conexiones establecidas con algún cliente realiza una pregunta de confirmación de la acción.
2) Change-log-level: permite cambiar el nivel de logueo de info según los niveles establecido: DEBUG - ACTIVIDAD- ERROR.
3) Show-connected-users: indica cuales son los usuarios conectados actualmente.

Ejemplo de archivo XML del servidor

```
<servidor>
    <CantidadMaximaClientes>8</CantidadMaximaClientes>
    <puerto>15636</puerto>
</servidor>
<usuarios>
    <usuario>
        <nombre>Juan</nombre>
        <password>123</password>
    </usuario>
    <usuarios>
        <nombre>Maria</nombre>
        <password>654</password>
    </usuarios>
</usuarios>
```

El cliente

Se deberá desarrollar un programa en C/C++ que lea desde un archivo en formato XML la definición de la dirección IP en la cual se encuentra ejecutando el servidor y el puerto en el cual está escuchando. Además este archivo posee una lista de mensajes a ser procesados. El nombre del archivo será pasado por línea de comandos.

Una vez que el cliente inicia, este creará un menú por pantalla según corresponda, en el cual se podrá seleccionar una de las siguientes opciones:

1. **Conectar**: Se establecerá la conexión con el servidor.
2. **Desconectar**: Se cerrará la conexión con el servidor.
3. **Salir**: Se finalizará la ejecución del programa.
4. **Login**: se pide usuario y contraseña para iniciar el chat.
5. **StressTest**: permite el envío de múltiples mensajes en forma iterativa durante una cantidad determinada de milisegundos que deberá ser ingresado tras la selección de esta opción. Los mensajes serán provistos a través de un archivo de texto especificado en el archivo xml de configuración del cliente.

6. **Revisar Buzón**: Puede observar todos los mensajes que dejaron en su buzón de mensajes personal.
7. **Mensaje Chat**: Enviar mensaje a conversación: Se envía un mensaje al grupo de chat.
8. **Mensaje Privado**: Se envía un mensaje a un destinatario particular.

Al igual que el programa llamado servidor el cliente deberá informar los distintos eventos en un archivo de log. Este archivo además debe tener información sobre errores de comunicación, pérdida de mensajes, mensajes recibidos, etc.

Ejemplo de archivo XML del Cliente

```
<cliente>
<conexion>
        <IP>192.168.0.1</IP>
        <puerto>65532</puerto>
</conexion>
<testfile>
        <path>/tmp/filetest_01</path>
</testfile>
</cliente>
```

Restricciones

- La implementación deberá estar hecha en C/C++.
- Para la lectura y escritura de archivos XML debe utilizarse una biblioteca.
- Todo el código debe ser desarrollado íntegramente por cada grupo. No se permite la reutilización de código de cuatrimestres anteriores o de otras materias. Ante cualquier duda se deberá consultar con los docentes. La reutilización de código sin consulta previa será condición suficiente para la desaprobación de la materia.

Fechas

(Este cronograma puede sufrir modificaciones)

Semana #	Fecha	Tema
1	16 de agosto	Presentación de la materia
2	**23 de agosto**	**Presentación enunciado TP Fase 1**
3	30 de agosto	Consultas y estado de avance de TP Fase 1.
4	6 de septiembre	Consultas y estado de avance de TP Fase 1.
5	13 de septiembre	Consultas y estado de avance de TP Fase 1.
6	**20 de septiembre**	**Primer Entrega TP Fase 1.** **Presentación enunciado TP Fase 2**
7	27 de septiembre	Segunda Entrega TP Fase 1. Consultas y estado de avance de TP Fase 2.
8	4 de octubre	Tercera Entrega TP Fase 1. Consultas y estado de avance de TP Fase 2.
9	11 de octubre	Consultas y estado de avance de TP Fase 2.
10	**18 de octubre**	**Primer Entrega TP Fase 2.** **Presentación enunciado TP Fase 3**
11	25 de octubre	Segunda Entrega TP Fase 2. Consultas y estado de avance de TP Fase 3.
12	1 de noviembre	Tercera Entrega TP Fase 2. Consultas y estado de avance de TP Fase 3.
13	8 de noviembre	Consultas y estado de avance de TP Fase 3.
14	15 de noviembre	Consultas y estado de avance de TP Fase 3.
15	**22 de noviembre**	**Primer Entrega TP Fase 3.**
16	29 de noviembre	Segunda Entrega TP Fase 3.
17	6 de diciembre	Tercera Entrega TP Fase 3.