Universidad del Valle de Guatemala
Sistemas Operativos
Semestre 1, 2020
Docente: Tomás Gálvez

Proyecto: chat

Este proyecto está basado en el propuesto por Bob Dugan y Erik Véliz en 2006. Con este trabajo se reforzarán los conocimientos sobre procesos, *threads*, concurrencia y comunicación entre procesos. El objetivo es desarrollar una aplicación de chat en C/C++. Para desarrollar el proyecto se recomienda leer https://linux.die.net/man/7/ip y sus enlaces relacionados, así como investigar el uso de SOCKETS.

La aplicación debe estar separada dos partes:

Servidor: mantiene una lista de todos los clientes/usuarios conectados al sistema. Sólo puede existir un servidor durante una ejecución del sistema de chat. El servidor se podrá ejecutar en cualquier máquina Linux que provea la infraestructura necesaria y los clientes se conectarán al servidor mediante la dirección IP de la máquina donde se ejecute el servidor en la red local.

Cliente: se conecta y se registra con el servidor. Un sistema de chat permite la existencia de uno o más clientes durante su ejecución. Posee una pantalla mediante la que se despliegan los mensajes que el usuario ha recibido del servidor, enviados por un usuario remoto; y en la que se permite el ingreso de texto para envío de mensajes a otros usuarios. Cada cliente debe presentar su propia y única pantalla de chat.

El servidor se ejecuta como un proceso independiente mediante el siguiente comando:

`<nombredelservidor> <puertodelservidor>`

Donde `<nombredelservidor>` es el nombre del programa y `<puertodelservidor>` es el número de puerto (de la máquina donde se ejecuta el servidor) en el que el servidor está pendiente de conexiones y mensajes de sus clientes. El servidor debe atender las conexiones de sus clientes con *multithreading*, usando el modelo propuesto en el libro/presentación de ese tema. El servidor no permitirá que existan dos usuarios conectados con el mismo nombre.

El servidor deberá proveer los siguientes servicios:

- Registro de usuarios: el cliente envía al servidor su nombre de usuario. El servidor registra el nombre de usuario junto con la dirección IP de origen en la lista de usuarios conectados *si ni el nombre ni la dirección existen ya en su registro*. De lo contrario, el servidor envía una respuesta que el cliente usa para desplegar un mensaje de error. Al aceptar la conexión del cliente se debe crear su sesión en el servidor y manejarla de forma concurrente a las demás sesiones y a la recepción de nuevas conexiones (*ergo*, *multithreading*).
- Liberación de usuarios: cuando un usuario cierra su cliente de chat o elige salir, el procedimiento debe conllevar aviso al servidor (o detección por parte de éste) para el cierre controlado de la sesión asociada. Esto es: que el servidor remueva la información del cliente del listado de usuarios conectados y, consecuentemente, que niegue intentos futuros de comunicación con ese usuario hasta que se registre un nuevo usuario con ese nombre, sin errores.
- Listado de usuarios conectados: el cliente podrá solicitar al servidor, en cualquier momento, el listado de usuarios conectados. El servidor responderá con el listado de nombres de usuarios conectados.

- Obtención de información de usuario: durante su ejecución, un cliente puede solicitar información de un usuario conectado en específico. La solicitud indica al servidor el nombre de usuario deseado y el servidor responde con la información asociada con ese nombre, si dicho nombre está conectado al servidor.
- Cambio de *status*: el cliente puede presentar cualquiera de los *status* siguientes: ACTIVO, OCUPADO e INACTIVO. ACTIVO es el *status* por defecto, pero un cliente, durante su ejecución, puede solicitar al servidor que cambie su *status*. El cliente reflejará el *status* que el servidor le tenga asignado siempre. El *status* INACTIVO puede ser asignado por el servidor a un cliente que ha pasado un período de inactividad predeterminado (establecido por l@s desarrollador@s). Procuren que el tiempo sea corto o modificable para facilitar la evaluación.
- *Boradcasting* y mensajes directos: los usuarios podrán comunicarse mediante el "chat general", donde el servidor envía los mensajes a todos los usuarios conectados, o enviarse mensajes directos, de un usuario específico a otro.

El cliente se ejecutará en mediante el siguiente comando:

`<nombredelcliente> <nombredeusuario> <IPdelservidor> <puertodelservidor>`

`<nombredelcliente>` es el nombre del programa. `<IPdelservidor>` y `<puertodelservidor>` serán a donde debe llegar la solicitud de conexión del cliente según la configuración del servidor.

El cliente podrá implementar la interfaz que el desarrollador desee, pero deberá proveer al usuario las facilidades para:

1. Chatear con todos los usuarios (*broadcasting*).
2. Enviar y recibir mensajes directos, privados, aparte del chat general.
3. Cambiar de *status*.
4. Listar los usuarios conectados al sistema de chat.
5. Desplegar información de un usuario en particular.
6. Ayuda.
7. Salir.

Para chatear con usuarios el formato debe ser similar a lo siguiente: `<usuario> <mensaje>`

Donde `<usuario>` es el destinatario a quien se entregará el `<mensaje>`. Cambiar de *status* permitirá al cliente elegir entre ACTIVO, OCUPADO e INACTIVO, y la elección enviará una solicitud de actualización de información al servidor. El cliente deberá refrescar su *status* una vez que el servidor haya realizado el cambio. El listado de usuarios se desplegará como especificado anteriormente.

Universidad del Valle de Guatemala
Sistemas Operativos
Semestre 1, 2020
Docente: Tomás Gálvez

Instrucciones de entrega:

- El proyecto debe entregarse, a más tardar, el 20 de abril de 2020, en el horario de clase.
- El proyecto se debe desarrollar en parejas o tríos. Cada grupo producirá un sistema de chat completo (servidor y cliente).
- El proyecto debe funcionar, como mínimo, en cualquiera de las máquinas virtuales OSC-2016 y OS-Concepts (pero debería funcionar también en sabores populares como Ubuntu).
- Todas las parejas deben presentar el mismo día, presencialmente. Todo trabajo (código, documentación, definiciones de protocolo, investigaciones realizadas, etc.) debe estar subido en Canvas antes de la fecha y hora de la presentación. No se otorgará calificación a grupos que no suban su trabajo a Canvas.
- Cada grupo encenderá su servidor en una red local y los clientes de los demás grupos deberán poder conectarse a, e interactuar con, el servidor encendido. Se evaluará, entre otros, que los clientes de diferentes grupos puedan comunicarse mediante cualquier servidor encendido (ver distribución de puntos al final de esta página).
- Para armar la red local se necesitará un punto de acceso físico (*router*). El Departamento de Ciencia de la Computación dispone de escasas unidades funcionales. Se recomienda a cada grupo conseguir un *router* para probar su proyecto tanto durante el desarrollo como en la entrega. También se podrá solicitar el equipo disponible al Departamento, pero sujeto a cola y con duración del préstamo dependiente del tamaño de la cola. Se agradece anticipadamente a cualquier grupo que ofrezca un punto de acceso para pruebas de su proyecto y los de los demás durante el día de presentación.
- **Es necesario acordar, entre toda la clase, un protocolo de comunicación entre clientes y servidores para garantizar la conectividad e interacción entre clientes y servidores de diferentes grupos. Definir el protocolo conlleva establecer los pasos con los que se inicia y termina una conexión, así como el formato (componentes y orden) de cada comunicación entre el cliente y servidor. Se abrirá una actividad en Canvas para entrega del protocolo, con fecha límite 08 de marzo, antes de medianoche. Cualquier miembro de la clase podrá hacer la entrega, siempre que tenga la aprobación de tod@s l@s demás alumn@s.**

Componente		%
Cliente	Chateo general con usuarios	10
	Chateo privado con *multithreading*	10
	Cambio de *status*	5
	Listado de usuarios e información de un usuario	5
Servidor	Atención con *multithreading*	10
	Broadcasting y mensajes directos	10
	Registro de usuarios	5
	Liberación de usuarios	5
	Manejo de *status*	5
	Respuesta a solicitudes de información de usuario(s)	5
Comunicación con otros proyectos		30
TOTAL		**100**