Reto 1: Sistema de Chat con Colas de Mensajes

Por: Miguel Villegas Nicholls, Juan José Restrepo Cardona y Esteban Molina Mejia.

1. Introducción

En el contexto actual, la comunicación instantánea es una necesidad fundamental en la vida cotidiana. Aplicaciones como WhatsApp o Telegram son ejemplos claros de cómo los sistemas de mensajería soportan interacciones dinámicas entre múltiples usuarios.

Este proyecto propone la implementación de un sistema de chat simple entre procesos, utilizando colas de mensajes como mecanismo de comunicación entre procesos (IPC, Inter-Process Communication). El objetivo es simular un servidor de chat centralizado que soporte múltiples salas, donde varios clientes puedan unirse, enviar y recibir mensajes en tiempo real.

2. Objetivos del Proyecto

- Desarrollar un sistema de chat que permita la comunicación entre múltiples procesos en C.
- Aplicar colas de mensajes como mecanismo de IPC en Linux.
- Soportar la creación y gestión de múltiples salas de chat.
- Implementar un servidor central que administre salas, usuarios y reenvío de mensajes.
- Desarrollar clientes capaces de unirse a salas, enviar mensajes y recibir mensajes de otros usuarios.

3. Requisitos Técnicos

- Lenguaje: C
- Mecanismo de IPC: Colas de mensajes (System V o POSIX)
- Sistema Operativo: Linux
- Trabajo en equipo: 3 integrantes
- Tiempo estimado de desarrollo: 10 días

4. Arquitectura del Sistema

El sistema está compuesto por dos tipos de procesos:

1. Servidor Central
    a. Gestiona las colas de mensajes globales y específicas de cada sala.
    b. Mantiene la lista de salas activas y los usuarios que pertenecen a cada una.
    c. Reenvía los mensajes enviados por un cliente al resto de usuarios de la sala correspondiente.
    - Aquí se usan funciones como msgget() para crear la cola global y las colas por sala.
    - Se hace msgrcv() constantemente para leer los mensajes entrantes de los clientes.
    - Al recibir un JOIN, el servidor revisa si la sala existe o crea una nueva.
    - Se recibe el mensaje con msgrcv() y se reenvía a cada usuario de la lista con msgsnd().

2. Clientes
    a. Representan a los usuarios del chat.
    b. Se conectan al servidor mediante la cola global.
    c. Pueden solicitar unirse a una sala existente o crear una nueva.
    d. Envían mensajes al servidor, que los reenvía a la sala.
    e. Escuchan constantemente en la cola de su sala para recibir mensajes de otros usuarios.

- Representan a los usuarios del chat
    - Cada cliente corre como un proceso independiente.
    - Se conectan con msgget() a la cola global para mandar solicitudes al servidor.
- Solicitan unirse a una sala o crearla
    - Mandan un mensaje con un campo tipo = JOIN.
    - El servidor contesta con el id de la cola de la sala.
- Envían mensajes al servidor
    - El cliente usa msgsnd() para escribir mensajes en la cola de la sala.
- Escuchan mensajes de otros usuarios
    - Se implementa un bucle con msgrcv() que imprime los mensajes recibidos en consola.

5. Flujo de Comunicación

5.1 Inicialización

- El servidor crea una cola global para recibir solicitudes. (msgget(CLAVE_GLOBAL, IPC_CREAT | 0666);)

```
emolinam@ThinkPad-Esteban23:/mnt/c/Users/Esteban Molina/OneDrive - Universidad EAFIT/Escritorio/EAFIT/SEMESTRE 6/Sistemas Operativos/Sistemas-Operativos/Parcial2$ ./servidor
=== SERVIDOR DE CHAT INICIADO ===
Cola global ID: 0
Nueva sala creada: 'General' con cola ID 1
Nueva sala creada: 'Deportes' con cola ID 2
Esperando clientes... (Ctrl+C para salir)
```

- Los clientes inician enviando un mensaje al servidor para registrarse. (Cliente: envía un REGISTER al servidor con su nombre de usuario.)

```
emolinam@ThinkPad-Esteban23:/mnt/c/Users/Esteban Molina/OneDrive - Universidad EAFIT/Escritorio/EAFIT/SEMESTRE 6/Sistemas Operativos/Sistemas-Operativos/Parcial2$ ./cliente Miguel
Bienvenido, Miguel. Salas disponibles: General, Deportes
```

5.2 Unirse a una sala

- El cliente envía un mensaje de tipo JOIN al servidor indicando la sala deseada.

```
emolinam@ThinkPad-Esteban23:/mnt/c/Users/Esteban Molina/OneDrive - Universidad EAFIT/Escritorio/EAFIT/SEMESTRE 6/Sistemas Operativos/Sistemas-Operativos/Parcial2$ ./cliente Miguel
Bienvenido, Miguel. Salas disponibles: General, Deportes
> join deportes
Te has unido a la sala: deportes
```

- El servidor crea la cola de la sala (si no existe) y agrega al cliente a esa sala.

```
[JOIN] Miguel -> deportes:
Nueva sala creada: 'deportes' con cola ID 3
Usuario 'Miguel' (PID: 449) agregado a sala 'deportes' (1/20)
```

- El servidor responde al cliente con el identificador de la cola de la sala.

```
Usuario 'Miguel' (PID: 449) agregado a sala 'deportes' (1/20)
```

5.3 Enviar un mensaje

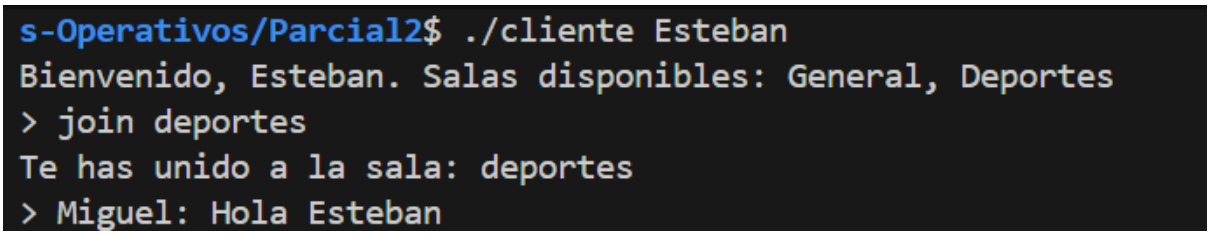
- El cliente escribe un mensaje en la sala.

```
emolinam@ThinkPad-Esteban23:/mnt/c/Users/Esteban Molina/OneDrive - Universidad EAFIT/Escritorio/EAFIT/SEMESTRE 6/Sistemas Operativos/Sistemas-Operativos/Parcial2$ ./cliente Miguel
Bienvenido, Miguel. Salas disponibles: General, Deportes
> join deportes
Te has unido a la sala: deportes
> hola
hola
```

- El mensaje se envía a la cola de esa sala.

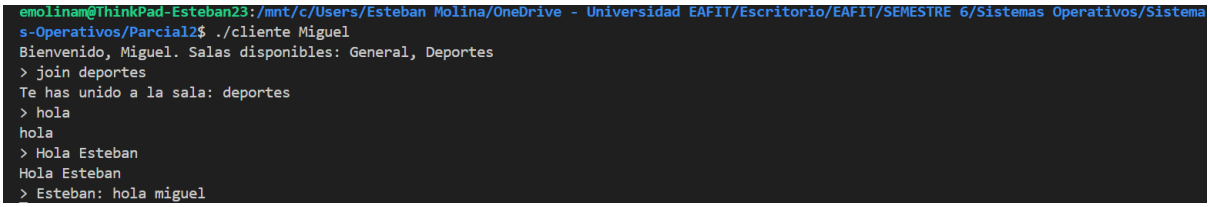
```
[CHAT] Miguel -> deportes: hola
```

- El servidor lo recibe y lo reenvía a todos los miembros de la sala, excepto al remitente.

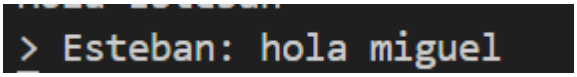


5.4 Recibir mensajes

- **Cada cliente escucha en su cola de sala.**



- **Al recibir un mensaje, lo despliega en pantalla con el formato:**



remitente: mensaje

6. Conclusiones

- **El proyecto permitió aplicar el uso de colas de mensajes en C como un mecanismo real de comunicación entre procesos.**
- **Se evidenció la importancia de un servidor central para la gestión de múltiples clientes y salas.**
- **El sistema implementado sienta las bases para ampliar funcionalidades como historial de mensajes, autenticación de usuarios o persistencia de salas.**