**Taller 6**



**Juan David Moran Santiusty**

**Andrés Felipe Sandino**

**Taller número 6 en el curso Laboratorio Sistemas Operativos**

**Profesor:**

**PABLO A. MAGÉ I.**

**Universidad del Cauca**

**Facultad de Ingeniería Electrónica y Telecomunicaciones**

**Departamento de Sistemas**

**Laboratorio Sistemas Operativos**

**Popayán, abril 2025**

**Juan David Mora Santiusty**

**Andrés Felipe Sandino**

**Taller 6**

Taller No. 6 presentado en el curso de Laboratorio Sistemas Operativos

Estudiantes del:

Programa de Ingeniería de Sistemas

Profesor:

Pablo A. Magé I

Popayán

2025

Contenido

[Contenido i](#_Toc128686608)

[Lista de Figuras ii](#_Toc128686609)

[Título del trabajo 1](#_Toc128686612)

[1.- Enunciado 1](#_Toc128686613)

[2.-Pseudocódigo](#_Toc128686616) 2

3[.-Descripción archivo Makefile](#_Toc128686616) 5

4[.-Resultados](#_Toc128686616) 6

# Lista de Figuras

Figura 3.1. Makefile5

Figura 4.1. Ejecución6

Figura 4.2. Log6

# 1.- Enunciado

Se deberá desarrollar dos programas en C que implementen la funcionalidad de conectar un servidor con uno cliente, a través de sockets de dominio AF\_UNIX (Basarse en el código fuente ej-unixsocket.rar). El programa servidor recibirá conexiones y creará el socket en una ruta especificada por argumentos de línea de comandos.

./unix\_server <socket-path>

Por ejemplo, si ruta del socket será: /tmp/unix\_socket el comando sería:

./unix\_server /tmp/unix\_socket

El programa cliente recibirá por la línea de comandos: la ruta del socket donde el servidor creará el socket a conectarse.

./unix\_client <socket-path>

Por ejemplo, si ruta del socket es: /tmp/unix\_socket el comando sería:

./unix\_client /tmp/unix\_socket

Una vez que el proceso cliente y el proceso servidor se encuentren conectados, primero el cliente enviará mensajes al servidor, quien deberá recibirlo y guardarlo en memoria (almacenarlo en un arreglo). Cuando el cliente ingresa el comando /exit, se cierra el socket y finaliza el proceso cliente.

El proceso servidor debe responder adecuadamente a las señales SIGTERM, cuando se reciba la señal el servidor deberá crear un archivo (./ruta/log-servidor.txt) en donde se almacenará la fecha y hora de inicio y terminación del proceso servidor.

El entorno de compilación y ejecución de los procesos cliente y servidor debe realizarse en una máquina virtual con sistema operativo Debian 12.

# 2.-Pseudocódigo

El siguiente pseudocódigo muestra la lógica para manejar la parte del servidor, empieza recibiendo la ruta en donde se creara el Socket y declarando las diferentes variables para su manejo, configura el manejador de señales y crea el socket asegurándose de finalizar la ejecución si la creación de este tuvo algún error, cuando la creación y el enlace al socket se ejecuten sin problema se esperara a la llegada de algún cliente, una vez este llegue se recibirán los mensajes del cliente imprimiéndolos hasta que el cliente finalice su ejecución finalmente el servidor esperar a la señal para terminar con su ejecución.

PROCEDIMIENTO create\_unix\_server(SOCKET\_PATH)

DECLARAR server\_fd, client\_fd COMO ENTEROS

DECLARAR server\_addr COMO STRUCT sockaddr\_un

CONFIGURAR manejador de señal SIGTERM

INICIAR hora\_inicio CON tiempo\_actual

CREAR socket UNIX Y ASIGNAR A server\_fd

SI server\_fd == -1 ENTONCES

MOSTRAR "Error al crear socket del servidor"

FINALIZAR EJECUCIÓN

FIN SI

ELIMINAR (unlink) CUALQUIER SOCKET EXISTENTE EN SOCKET\_PATH

INICIALIZAR estructura server\_addr

ASIGNAR familia de direcciones COMO AF\_UNIX

COPIAR SOCKET\_PATH A server\_addr.sun\_path

ENLAZAR (bind) server\_fd CON server\_addr

SI falla ENLACE ENTONCES

MOSTRAR "Error en bind"

FINALIZAR EJECUCIÓN

FIN SI

ESCUCHAR conexiones en server\_fd

SI falla ESCUCHA ENTONCES

MOSTRAR "Error en listen"

FINALIZAR EJECUCIÓN

FIN SI

MOSTRAR "Servidor UNIX esperando conexiones..."

ACEPTAR una conexión Y ASIGNAR A client\_fd

SI falla ACEPTAR ENTONCES

MOSTRAR "Error en accept"

FINALIZAR EJECUCIÓN

FIN SI

MIENTRAS mensaje\_recibido NO SEA "/exit"

LEER mensaje del cliente

MOSTRAR mensaje recibido

FIN MIENTRAS

MOSTRAR "Cliente cerró la sesión"

CERRAR client\_fd

MOSTRAR "Esperando SIGTERM para finalizar..."

ESPERAR a señal SIGTERM

CERRAR server\_fd

ELIMINAR (unlink) SOCKET\_PATH

FIN PROCEDIMIENTO

El siguiente pseudocódigo muestra la lógica del cliente que empieza recibiendo la ruta del socket al que se conectara y declarando las variables para manejar la conexión al socket, una vez la conexión se ejecute sin problema se espera a que se ingrese alguna palabra en la linea de comandos para enviarle la palabra al servidor sigue haciendo esto hasta que se ingrese /exit una vez se ingrese esta palabra se finalizara la ejecución del cliente.

PROCEDIMIENTO create\_unix\_client(SOCKET\_PATH)

DECLARAR client\_fd COMO ENTERO

DECLARAR server\_addr COMO STRUCT sockaddr\_un

CREAR socket UNIX Y ASIGNAR A client\_fd

SI client\_fd == -1 ENTONCES

MOSTRAR "Error al crear socket del cliente"

FINALIZAR EJECUCIÓN

FIN SI

INICIALIZAR estructura server\_addr

ASIGNAR familia de direcciones COMO AF\_UNIX

COPIAR SOCKET\_PATH A server\_addr.sun\_path

CONECTAR client\_fd CON server\_addr

SI falla CONEXIÓN ENTONCES

MOSTRAR "Error en connect"

FINALIZAR EJECUCIÓN

FIN SI

DECLARAR mensaje COMO CADENA

MIENTRAS mensaje\_recibido NO SEA "/exit"

LEER mensaje desde el estándar de entrada

ENVIAR mensaje al servidor

FIN MIENTRAS

CERRAR client\_fd

FIN PROCEDIMIENTO

El ultimo pseudocódigo muestra la lógica de como manejar las señales SIGTERM, al recibir la señal de finalización se crea un archivo para guardar la hora de inicio y fin del servidor para al final finalizar la ejecución.

PROCEDIMIENTO manejador\_sigterm(sig)

INICIALIZAR log\_num COMO 1

ABRIR archivo "log-servidor.txt" en modo lectura Y ASIGNAR A temp

SI temp NO ES NULO ENTONCES

MIENTRAS haya líneas en temp HACER

LEER línea del archivo

SI la línea contiene "[LOG" ENTONCES

INCREMENTAR log\_num

FIN SI

FIN MIENTRAS

CERRAR temp

FIN SI

ABRIR archivo "log-servidor.txt" en modo adjuntar Y ASIGNAR A archivo

SI archivo ES NULO ENTONCES

MOSTRAR "No se pudo abrir el archivo de log"

FINALIZAR EJECUCIÓN

FIN SI

OBTENER hora\_fin COMO tiempo actual

ESCRIBIR en archivo el siguiente contenido:

- Número de log: log\_num

- Hora de inicio del servidor

- Hora de finalización del servidor

CERRAR archivo

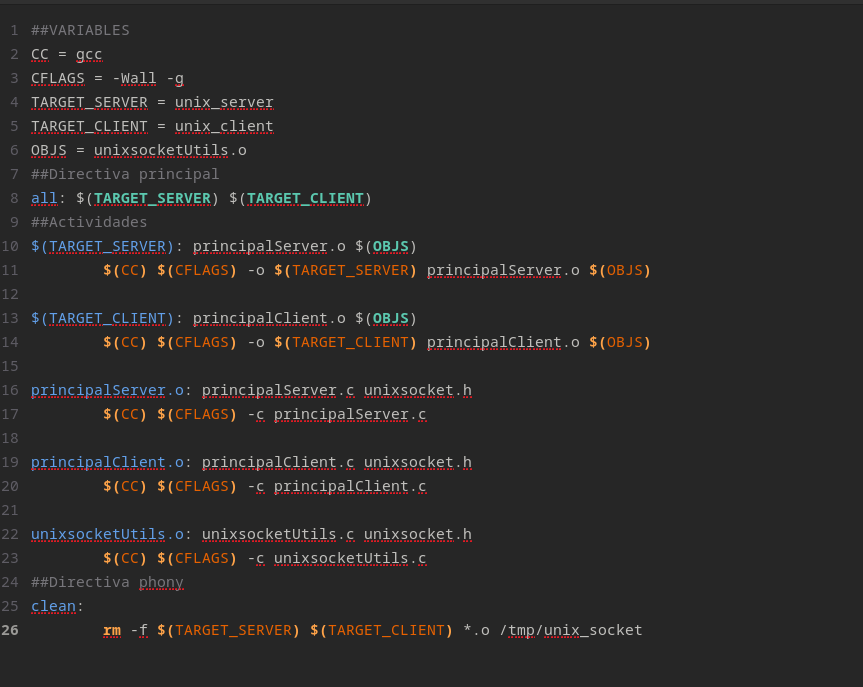
MOSTRAR "[LOG] Señal SIGTERM recibida. Archivo de log generado."

FINALIZAR EJECUCIÓN

FIN PROCEDIMIENTO

# 3.- Descripción Archivo Makefile

El archivo Makefile se encarga de la creación del dos ejecutables, el ejecutable para el servidor y para el cliente en ambos casos utilizando el archivo unixsocketUtils.c además de la instrucción clean para eliminar los ejecutables y archivos .o además de eliminar el socket creado en la ejecución.

Figura 3.1. Makefile

# 4.- Resultados

Para probar el funcionamiento de la solución usaremos la ruta /tmp/unix\_socket en el ejecutable y mostraremos como el servidor recibe los mensajes del cliente hasta que lleguen las señales para finalizar la ejecución.

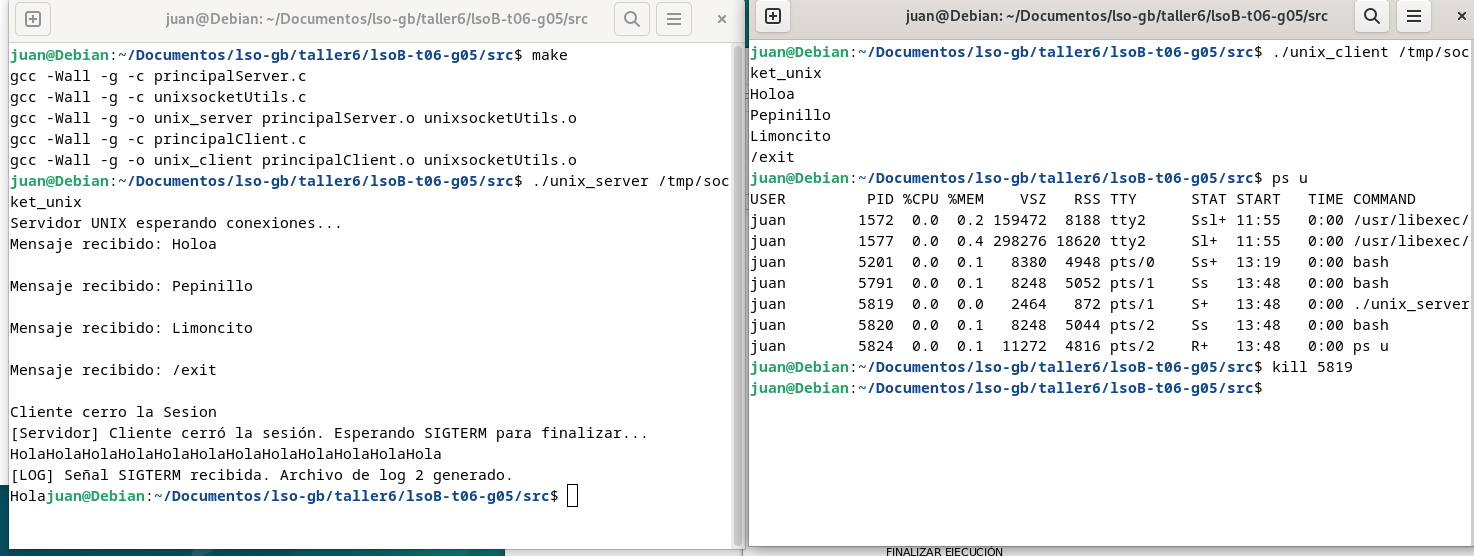


Figura 4.1. Ejecución

Figura 4.2. Log

Bibliografía

[1]

“Taller 06: Comunicación IPC-Sockets UNIX-señakes,” Popayan, Apr. 2025. [Online]. Available: www.unicauca.edu.co