Instituto Politécnico Nacional

IPN

Escuela Superior de Cómputo

ESCOM

Academia de Redes Aplicaciones para comunicaciones en red

6CV2

Práctica 15

“Ejemplo Protocolo HTTP”

Alumna:

Navarrete Becerril Sharon Anette

Fecha de entrega: “ 25 - Noviembre - 2024”

Profesor: Ojeda Santillan Rodrigo

OBJETIVO

Desarrollar un servidor HTTP básico en lenguaje C para ilustrar el funcionamiento fundamental del protocolo HTTP, que constituye la base de la comunicación en la web. A través de este ejercicio, se busca demostrar cómo un servidor en C puede aceptar conexiones en el puerto 80, procesar solicitudes de clientes y responder con una página HTML simple, proporcionando una introducción a la construcción de servidores web en un entorno de bajo nivel.

INTRODUCCIÓN

El Protocolo de Transferencia de Hipertexto (HTTP) es un protocolo de la capa de aplicación que establece las reglas para la comunicación entre clientes, como navegadores, y servidores web. Este protocolo es fundamental en la infraestructura de la web, ya que permite que los clientes soliciten y reciban contenido estructurado, como páginas HTML, imágenes y otros recursos. En este ejemplo, se utiliza el lenguaje de programación C para construir un servidor HTTP básico que escucha en el puerto 80 y procesa las solicitudes entrantes.

Mediante el uso de sockets TCP, que aseguran una comunicación confiable y orientada a la conexión, el servidor implementado en C acepta conexiones de clientes, lo que permite enviar respuestas HTTP estructuradas en la forma de una página HTML con un mensaje de bienvenida y un botón interactivo. Este ejemplo también destaca cómo un servidor escrito en C puede gestionar solicitudes HTTP a bajo nivel, lo que resulta ideal para comprender la mecánica de transmisión de datos estructurados en la web y la manipulación directa de protocolos de comunicación.

DESARROLLO

Para implementar el servidor HTTP en C, se inicia configurando un socket TCP que permita la comunicación con los clientes. Este socket se enlaza al puerto 80, el puerto estándar de HTTP, y se configura para escuchar conexiones entrantes. Una vez que el servidor está en funcionamiento, entra en un bucle de espera, aceptando conexiones de clientes que intentan acceder a la página.

Cada vez que se establece una conexión, el servidor procesa la solicitud y responde con una página HTML simple. Este contenido HTML contiene un mensaje de bienvenida y un botón, demostrando que el servidor puede responder con elementos interactivos. Para manejar esta interacción, el servidor envía la respuesta HTTP de manera estructurada, asegurándose de que el cliente reciba los datos correctamente formateados según los estándares de HTTP.

Para lograr este funcionamiento se debe de agregar el siguiente fragmento de código al archivo de ejemplohttp15.c :

```
#include <stdio.h>
#include <stdlib.h>
```

```
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>

int udp_socket, udp_cliente;
char Block[1000] = "";
struct sockaddr_in servidor, cliente;

int main()
{
    int lbind, llisten;

    udp_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (udp_socket == -1) {
        perror("\nError al tratar de abrir socket...");
        exit(-1);
    } else {
        perror("\nÉxito al abrir socket...");
        servidor.sin_family = AF_INET;
        servidor.sin_port = htons(80);
        servidor.sin_addr.s_addr = INADDR_ANY;
        lbind = bind(udp_socket, (struct sockaddr *)&servidor,
sizeof(servidor));
        if (lbind == -1) {
            perror("\nError en el bind");
exit(0);
        } else {
            perror("\nÉxito en bind");
            llisten = listen(udp_socket, 5);
            if (llisten == -1) {
                perror("\nError en el listen");
                exit(0);
            } else {
                perror("\nÉxito en el listen");
                sprintf(Block,
                        "HTTP/1.0 200 OK\r\n"
                        "Content-type: text/html\r\n"
                        "Content-length: %d\r\n\r\n"
                        "<!doctype html>"
                        "<html>"
                        "<head>"
                        "<title>HTTP</title>"
                        "</head>"
                        "<body bgcolor=\"pink\">" // Fondo rosa
                        "<h1>Prueba practica HTTP</h1>"
                        "<button type=\"button\"
onclick=\"document.getElementById('ejemplo').innerHTML=Date()\">"
                        "Fecha y hora"
                        "</button>"
                        "<p id=\"ejemplo\"></p>"
                        "</body>"
                        "</html>",
                        300);

                while (1) {
                    int size_cliente = sizeof(struct sockaddr_in);
                    udp_cliente = accept(udp_socket, (struct sockaddr
*)&cliente, &size_cliente);
```

```
                    if (udp_cliente == -1) {
                        perror("\nError en el accept");
                        exit(0);
                    } else {
                        perror("\nÉxito en el accept");
                        write(udp_cliente, Block, strlen(Block));
                        close(udp_cliente);
                    }
                }
                close(udp_socket);
                return 0;
            }
        }
    }
}
```

Para poder probar el código se debe de compilar de la siguiente manera:

```
gcc ejemplohttp.c -o ejemplohttp
```

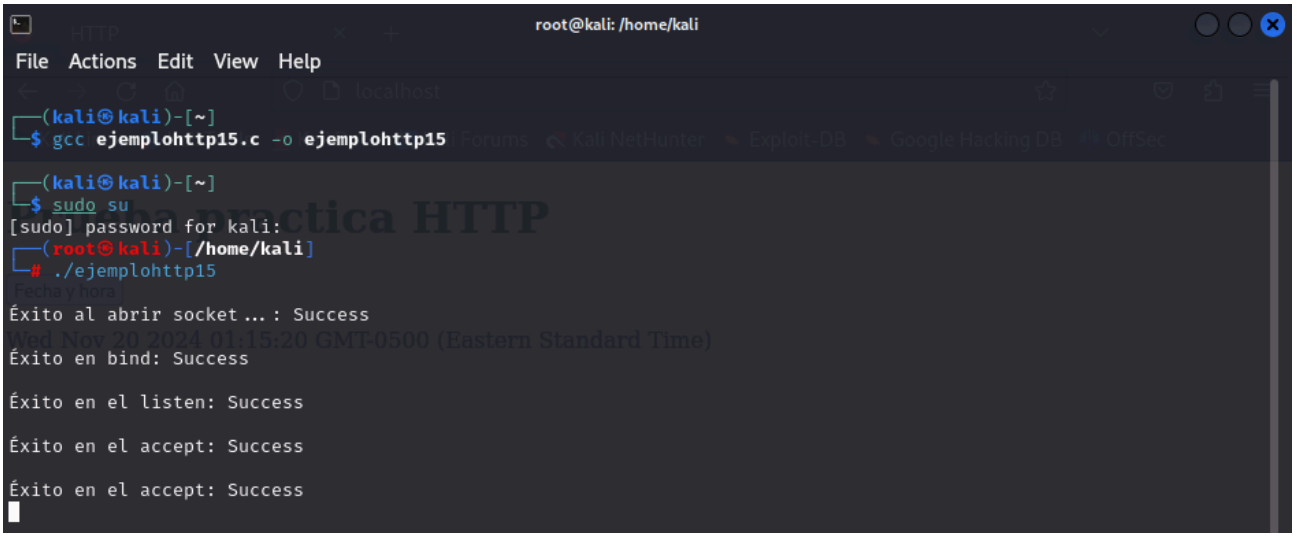
Suele denegar el acceso muchas veces por lo que recomiendo que en la terminal de Ubuntu se ingrese el siguiente comando:

```
Sudo su
```

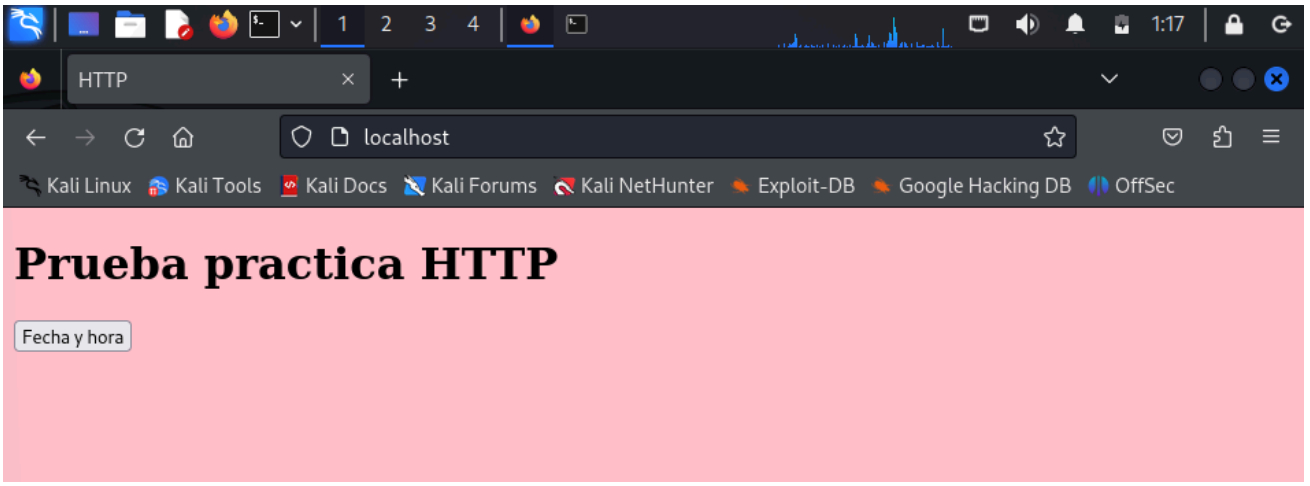
Posterior a ello ejecutar con:

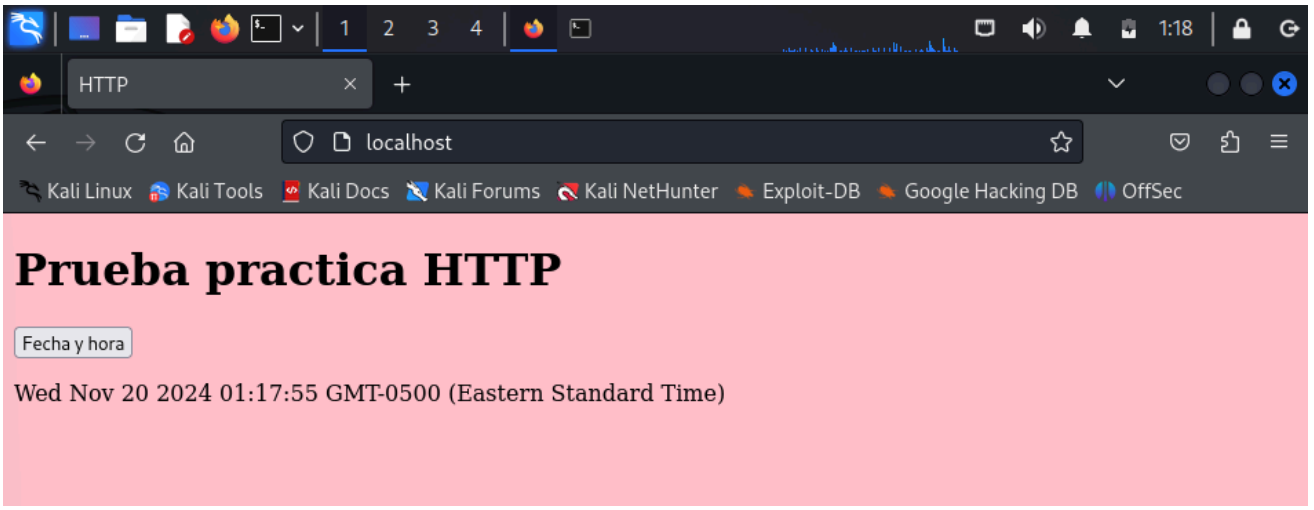
```
./ejemplohttp15
```

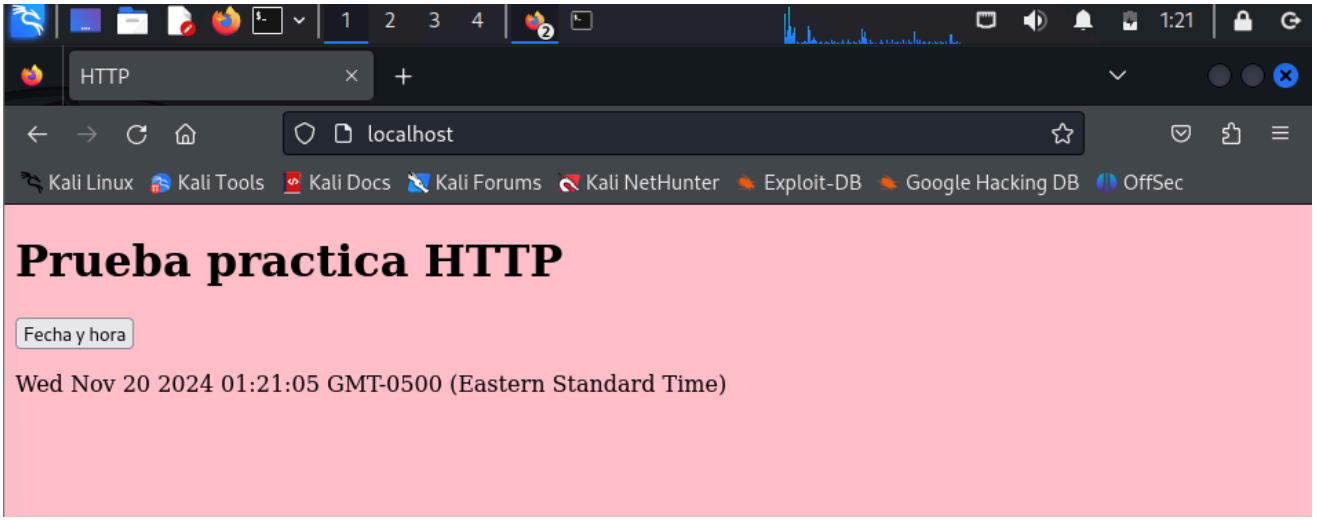
Captura de la ejecución del funcionamiento:



Al consultar en el navegador accediendo con localhost:80 podremos ver la página que se está invocando:







CONCLUSIONES

Navarrete Becerril Sharon Anette:

La implementación de un servidor HTTP básico en lenguaje C permite comprender los principios fundamentales del protocolo HTTP, así como el uso de sockets TCP para establecer una comunicación confiable entre cliente y servidor. Este ejercicio ilustra cómo los protocolos de nivel de aplicación funcionan a través de mecanismos de red de bajo nivel.

BIBLIOGRAFÍA

Stevens, W. Richard. Programación en red con Unix: La API de Sockets. Addison-Wesley Professional, 2003.

Forouzan, Behrouz A. Comunicaciones de Datos y Redes. McGraw-Hill, 2012. Tanenbaum, Andrew S., y David J. Wetherall. Redes de Computadoras. Pearson, 2010.

Comer, Douglas E. Internetworking con TCP/IP Volumen Uno. Prentice Hall, 2006. Kurose, James F., y Keith W. Ross. Redes de Computadoras: Un Enfoque Descendente. Pearson, 2017.

Beazley, David, y Brian K. Jones. Python Cookbook: Recetas para Dominar Python 3. O'Reilly Media, 2013. Begg, A. Sockets TCP/IP en C: Guía Práctica para Programadores. Morgan Kaufmann, 2000