Práctica 2.2 TCP/IP. CLIENTE SERVIDOR

Jordi Blasco Lozano

Sistemas operativos y distribuidos

Grado en Inteligencia Artificial

Indice:

1. Introducción

En esta práctica se ha implementado una aplicación cliente-servidor utilizando sockets TCP. El objetivo principal es que el servidor transfiera el archivo "Google.html" al cliente, quien posteriormente lo mostrará por pantalla. El servidor se mantiene a la escucha por el puerto 9999, aceptando conexiones y generando un proceso hijo para cada cliente que se conecte.

2. Trabajos a realizar

- Crear un servidor que pueda recibir conexiones y enviar un archivo al cliente.
- Crear un cliente que se conecte al servidor y reciba el archivo.

3. Implementación del servidor

La función principal del servidor es escuchar a los clientes y transferirles el archivo.

3.1 Creación del socket y enlace

Primero, se usa la función **socket**() para crear un socket. Luego se configura la dirección del servidor usando la estructura **sockaddr_in** y se conecta al socket con **bind**(). Después, usamos **listen**() para permitir que el servidor escuche hasta 5 conexiones.

3.2 Escucha y aceptación de conexiones

Con **accept**(), el servidor acepta conexiones de los clientes. Cada vez que se acepta una conexión, se crea un proceso hijo usando **fork**(). El proceso hijo se encarga de enviar el archivo, mientras que el proceso padre sigue esperando nuevas conexiones.

3.3 Transferencia de archivos

El archivo "Google.html" se abre y se lee en partes de tamaño BUFFERSIZE que se envían al cliente con **send**(). Cuando se termina de enviar el archivo, el proceso hijo cierra la conexión.

4. Implementación del cliente

El cliente tiene la funcion de conectarse al servidor, recibir el archivo y posteriormente escribirlo.

4.1 Creación del socket y conexión

El cliente usa **socket**() para crear el socket y luego **connect**() para conectarse al servidor. Para ejecutar el cliente, se necesita especificar la dirección IP del servidor que sera pasada por parametro.

4.2 Recepción de archivos

Una vez conectado, el cliente recibe el archivo en partes usando **recv**(). El contenido del archivo se imprime en pantalla para asegurarse de que todo se recibió correctamente.

5. Resultados

```
jordiblascolozano@MacBook-Pro prac2 % runUbuntu
root@1eeb9dd78c47:/workdir# ./cliente 127.0.0.1
bash: ./cliente: Permission denied
root@1eeb9dd78c47:/workdir# chmod +x cliente
root@1eeb9dd78c47:/workdir# ./cliente 127.0.0.1
Conectado al servidor 127.0.0.1 en el puerto 9999
<html dir="ltr" lang="es" class="" lazy-loaded="true"><head>
    <meta charset="utf-8">
    <title>Nueva pestaña</title>
    <style>
      body {
        background: #3C3C3C;
        margin: 0;
      }

      #backgroundImage {
        border: none;
        height: 100%;
        pointer-events: none;
        position: fixed;
        top: 0;
        visibility: hidden;
        width: 100%;
      }

      [show-background-image] #backgroundImage {
        visibility: visible;
      }
    </style>
  <cr-iconset name="cr20" size="20">
  <svg>
    <defs>

      <g id="block">
        <path fill-rule="evenodd" clip-rule="evenodd" d="M10 0C4.4
.58 5.58 2 10 2C11.85 2 13.55 2.63 14.9 3.69L3.69 14.9C2.63 13.55
```

```
jordiblascolozano@MacBook-Pro prac2 % docker exec -it 1eeb9
e8d8969d95e45ee7cb0e86b26eb8a5328132fa880 bash
root@1eeb9dd78c47:/workdir# ./servidor
Servidor escuchando en el puerto 9999...
Conexión aceptada desde 127.0.0.1
Archivo enviado correctamente a 127.0.0.1
```