LABORATORIO 1

CLIENTE ECHO Y SERVIDOR ECHO.

Propósito

Aprender a modificar y extender programas de aplicación que se comunican sobre Internet

Generalidades

Desarrolle, compile y testee un programa cliente *echo* y un programa servidor *echo*. Luego ejecute ambos programas en computadoras separadas, y verifique que se pueden comunicar.

Procedimientos y Detalles

1. Con el código de ejemplo provisto en el teórico, deberá crear dos programas: Un `servidor echo` y un `cliente echo`.

2. El `servidor echo` deberá escuchar indefinidamente por comunicaciones pero sólo puede atender una a la vez. Una vez establecida dicha comunicación, deberá replicar todo lo que el `cliente echo` le envíe.

3. El `client echo` deberá conectarse con el `servidor echo` y una vez establecida la comunicación, enviará mensajes al mismo, el cual se los responderá, enviando exactamente la misma información.

4. La comunicación termina cuando el cliente ingresa la palabra `exit` o bien debido a alguna combinación de teclas (usted deberá averiguar cuál termina la conexión en un socket)

5. Para testear el `servidor echo`, vamos a necesitar un único *número de aplicación*. Si múltiples grupos están usando el laboratorio, será necesario para cada servidor corriendo que se le sea asignado un único número. Ya sea que su profesor le asigne un número único o coordinando entre ustedes para elegir los siguientes valores comenzando en 2001, 2002, 2003, y así sucesivamente. Anotar el número abajo.

   Número asignado:

6. Realizar un test de loopback con el `servidor echo` y el `cliente echo` ejecutándose en una computadora como se explicó en clases. Use el nombre de computadora `localhost` y el número de aplicación que se le fue asignado en el paso anterior.

7. Testear el `cliente echo` y el servidor ejecutando el `cliente echo` en una computadora y el servidor en otra.

8. **Obligatorio:** Modificar el servidor para que mantenga un archivo de log conteniendo todos los datos que el servidor replica.

9. **Obligatorio:** El servidor debe presentar un diseño tal que acepte el contacto de un `cliente echo`, maneje dicho contacto, y luego de terminar con el mismo, espere por el contacto de otro `cliente echo`.