Laboratorio 3: Interfaz de Sockets Berkeley

En los sistemas operativos tipo *UNIX*, tales como *GNU/Linux* y *macOS*, la interfaz de programación de aplicaciones provista por el sistema operativo para hacer uso de la pila de red del sistema se conoce como la interfaz de sockets Berkeley o interfaz de sockets *BSD*[1].

1. Funciones de la Interfaz de *Sockets* BSD

La interfaz de *sockets* BSD define las siguientes funciones:

socket() crea un descriptor de archivo para un *socket*.

connect() conecta un *socket* creado con `socket()` con un servidor.

bind() enlaza un descriptor de archivo creado con `socket()` con una dirección IP.

listen() coloca un descriptor de archivo creado con `socket()` en modo de escucha pasiva.

accept() acepta una conexión entrante, creando un nuevo descriptor de archivo para representar esa conexión.

Adicionalmente, la interfaz de *sockets* Berkeley provee funciones para crear sockets de datagramas (UDP), sockets crudos y sockets de UNIX.

La interfaz también define una serie de estructuras de datos para representar direcciones IP y propiedades de una conexión por *sockets*.

2. El Protocolo Echo

El protocolo de capa de aplicación más sencillo que existe es el protocolo *echo*, el cual fue definido por John Postel en el RFC-862, disponible en `https://tools.ietf.org/html/rfc862`. Este protocolo consiste sencillamente en un servidor que retorna a cada cliente los bytes que recibe, tal cual como los recibe; básicamente una extensión a Internet del comando `echo` de *UNIX*.

El archivo `echo_server.c` contiene la implementación de un servidor echo sencillo que tiene soporte para atender a un solo cliente a la vez. Si múltiples clientes se conectan al servidor, estos quedarán bloqueados en espera de que el servidor los acepte y puedan comunicarse.

Abra el archivo `echo_server.c` e identifique los pasos necesarios para crear un *socket* TCP pasivo, o *socket* de servidor.

Para probar el servidor, cómpilelo con el siguiente comando:

```
gcc -std=c99 -o es echo_server.c -D_POSIX_C_SOURCE=200112L
```

Ejecute el servidor con el comando `./es`. Luego ejecute el siguiente comando en otra terminal:

```
telnet 127.0.0.1 9989
```

Este comando se conecta al servidor mediante una conexión TCP directa. Escriba cualquier texto en la terminal donde está ejecutando `telnet` y presione enter. Verá como el servidor repite al cliente cualquier dato que este le envíe. Pruebe ejecutando el comando `telnet` en otra terminal adicional y envíe lineas de texto al servidor en ambas terminales.

Para cerrar la conexión `telnet` debe presionar las teclas `ctrl+]` (control corchete cerrado) para entrar al modo de comandos de `telnet`[2] y luego ejecutar el comando close. Pruebe cerrando las conexiones telnet en el orden en que las creó.

[1] Este nombre se deriva del hecho de que esta interfaz fue inventada en los años 80 para la versión de *UNIX* conocida como *BSD* (*Berkeley Software Distribution*), desarrollada en la Universidad de California, Berkeley.

[2] Sabrá que está en el modo de comandos de `telnet` porque la terminal imprimirá el texto `telnet>`.

Coordine con su preparador para ejecutar el comando `telnet` anterior cambiando la dirección IP `127.0.0.1` por la dirección IP de la computadora de otro compañero donde se deberá estar ejecutando el servidor echo[3].

El archivo `echo_client.c` implementa un cliente *echo*. Abra el archivo y observe como se define un *socket* cliente para conectarse con un *socket* de servidor. Compile el cliente con el siguiente comando:

```
gcc -std=c99 -o ec echo_client.c -D_POSIX_C_SOURCE=200112L
```

Ejecute nuevamente el servidor y luego ejecute el cliente como `./ec`. Pruebe enviando distintas cadenas al servidor y termine el cliente presionando `ctrl+D` en la terminal. Luego ejecute el cliente de la siguiente forma para probar la conexión con la computadora de un compañero que esté ejecutando el servidor echo:

```
./ec DIRECCION_IP
```

3. Resolución de Nombres de Dominio

Para evitar tener que colocar la dirección IP del *host* al que queremos conectarnos, podemos hacer uso del nombre de dominio del mismo si existe. Para esto utilizamos la función `getaddrinfo()` para solicitar el servicio al resolvedor DNS del sistema operativo. Utilizando el comando `man getaddrinfo`, discuta con su preparador cuales son los parámetros recibidos por esta función.

Abra el archivo `getaddrinfo.c`. Examine el contenido del archivo y observe como se hace uso de la función `getaddrinfo()` para resolver un nombre DNS. Compile el código fuente con el siguiente comando:

```
gcc -std=c99 -o gai getaddrinfo.c -D_POSIX_C_SOURCE=200112L
```

Luego ejecute el programa generado de las siguientes maneras:

- `./gai`
- `./gai google.com`
- `./gai correo.ciens.ucv.ve`
- `./gai seed.bitcoin.sipa.be`

Como podrá ver, la función *getaddrinfo()* puede retornar todas las direcciones asociadas a un nombre de dominio en caso de que haya más de una.

[3]Si la computadora de su compañero rechaza las conexiones, pruebe ejecutando el comando `iptables -F` como usuario `root` para desactivar el *firewall* del sistema.