Virtualización y Sistemas Operativos Avanzados

Trabajo Práctico N^{o}2
Año 2019

SOCKETS Y RPC

Formato de entrega

Se deberá enviar un email a constanza.iquaglia@gmail.com con el asunto "VSOA TP2" y un archivo adjunto .tar.gz que contenga los directorios doc, sck y rpc:

- Documentación:
  - Nombre, apellido y correo electrónico de cada integrante.
  - Respuestas a las consignas planteadas. Incluir el código fuente con las aclaraciones que se consideren necesarias y las instrucciones para compilar y ejecutar las aplicaciones.
- Código fuente: Entregar dentro de una carpeta src todo el código fuente desarrollado. Utilizar rpc/src para RPC y sck/src para sockets.
- Makefile: Archivo de reglas para *make* que incluya los comandos necesarios para poder correr *make* y *make clean*. Los objetos se deberán crear en la carpeta rpc/obj y sck/obj.
- Archivo de usuarios y contraseñas: Incluir un archivo de usuarios y contraseñas para poder probar los programas generados. Esto se explica con mayor detalle en la consigna del trabajo práctico.

Consigna

Se deberá implementar una aplicación cliente/servidor que permita validar un usuario y contraseña en el servidor y obtener estadísticas.

Para los fines del trabajo práctico, la validación solo se realizará buscando un usuario y contraseña en un archivo. La contraseña se almacenará en texto plano.

El servidor deberá determinar si el usuario y contraseña enviados por el cliente se encuentran o no en el archivo. El cliente además podrá solicitar estadísticas acerca de la cantidad de validaciones realizadas.

La aplicación deberá desarrollarse utilizando sockets y RPC.

Formato cliente

```
rauth -u <user> -p <pass>
```

- -u: Indica que el parámetro que sigue es un nombre de usuario.

- -p: Indica que el parámetro que sigue es una contraseña.

El servidor responderá con alguno de los siguentes códigos, en base a los cuales el cliente debe mostrar un mensaje en pantalla:

- 0: Usuario y contraseña correctos.
- 1: Usuario o contraseña incorrectos.
- 2: Cualquier otro error.

```
rauth -s
```

- -s: Solicita estadísticas al servidor.

El servidor deberá retornar una estadística con los valores devueltos.

Formato archivo

El formato de archivo a utilizar es a libre elección. Debe quedar documentado y se deberá proveer junto con el còdigo fuente un archivo de ejemplo para poder realizar pruebas.

Guía práctica

Sockets

Se va a utilizar la librería de C sys/socket.h.

Pasos del servidor

1. Creación del socket
   int sockfd = socket(int domain, int type, int protocol)
   - domain: Dominio de comunicación: AF_INET (IPv4), AF_INET (IPv6).
   - type: Tipo de comunicación: SOCK_STREAM (TCP), SOCK_DGRAM (UDP).
   - protocol: Protocolo de internet. 0 es IP.
   - Retorna un descriptor del socket.
2. Setear las opciones del socket (opcional).
   *int sockfd = setsockopt(int sockfd, int level, int optname, const void *optval, socklen_t optlen)*
   - sockfd: Descripción del socket.
   - level: Nivel para el cual se setea la opción. SOL_SOCKET (opciones a nivel socket), IPPROTO_IP (opciones a nivel IP), IPPROTO_IPV6.
   - optname: Nombre de la opción. SO_REUSEADDR, SO_REUSEPORT: Se setean en 1 para permitir reutilizar IP y puerto en diferentes conexiones.
   - optval: Puntero a los datos de la opción.
   - optlen: Longitud de optval.
3. Bind
   *int bind(int sockfd, const struct sockaddr * addr, socklen_t addrlen)*
   - sockfd: Descriptor del socket.
   - addr: Estructura que contiene la dirección IP
   - addrlen: Longitud de addr
4. Listen
   int listen(int sockfd, int backlog)
   - sockfd: Descriptor del socket.
   - backlog: Define el valor máximo de conexiones pendientes para el socket.
5. Accept
   *int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);*

Pasos del cliente

1. Creación del socket: Es igual que para el servidor.
2. Conectarse al servidor
   *int connect(int sockfd, const struct sockaddr * addr, socklen_t addrlen)*

Comunicación entre cliente y servidor

1. Send. Envía el mensaje al otro extremo (cliente o servidor).
   *int connect(int sockfd, const struct sockaddr * addr, socklen_t addrlen)*
   - Retorna -1 si hubo error (detectado localmente), 0 si se envió
2. Read. Recibe el mensaje.
   *int connect(int sockfd, const struct sockaddr * addr, socklen_t addrlen)*
   - Retorna la cantidad de bytes leídos. 0 indica conexión cerrada. -1 indica error.

RPC

Vamos a usar la herramienta rpcgen. Esta herramienta genera el código C necesario para implementar RPC.

```
# rpcgen -C foo.x
```

foo.x es un archivo RPCL (Remote Procedure Call Language), similar a C que consiste en una lista de procedimientos con sus parámetros.

Se generan los archivos:

- foo_clnt.c (cliente)
- foo_svc.c (servidor)
- foo_xdr.c (XDR - comunicación externa)
- foo.h (Archivo header - compartido por cliente y servidor)

Generar cliente y servidor (códigos de ejemplo):

```
# rpcgen -C -Sc foo.x > fooclient.c
```

```
# rpcgen -C -Ss foo.x > fooservices.c
```

Compilación de los archivos

Servidor:

```
# gcc -o fooserver fooservices.c foo_svc.c foo_xdr.c
-lrpcsvc -lnsl
```

Cliente:

```
# gcc -o fooclient fooclient.c foo_clnt.c foo_xdr.c -lnsl
```

Bibliografía

- Código de ejemplo: https://www.geeksforgeeks.org/socket-programming-cc/
- https://www.ibm.com/support/knowledgecenter/en/SSLTBW/
- Man pages.