

Trabajo práctico I

Sockets en sistemas tipo UNIX

Introducción

Los sockets son una abstracción de comunicación entre procesos que, en un sistema tipo UNIX, se implementan en un descriptor de archivo, sobre el cual se envía o recibe información, al igual que como se lee o escribe un archivo [1].

Son una herramienta muy importante, uno de los pilares de la comunicación entre procesos, y sumamente utilizada en la mayoría de las aplicaciones de red.

Objetivo


El objetivo del presente trabajo práctico es que el estudiante sea capaz de diseñar e implementar un software que haga uso de la API de sockets del Sistema Operativo, implementando lo visto en el teórico y haciendo uso todos los conocimientos adquiridos en Ingeniería de Software y Sistemas Operativos I.

Desarrollo

Se pide que diseñe, implemente y testeé un software (desarrollado en C), que permita realizar la conexión, control y transferencia entre uno o más clientes con un servidor que posee el baash (Lab 2) de Sistemas Operativos I.

El programa que corre en el Cliente debe permitir conectarse a al Servidor de forma segura (orientado a conexión), que tiene abierto un puerto fijo (6020). El cliente, debe tomar un número de un puerto libre de su sistema operativo.

Debe implementarse un sistema de autenticación (usuario y password) de acceso al servidor. Todo el proceso de autenticación debe realizarse del lado del servidor.



A los fines de establecer la conexión, el usuario debe ejecutar el programa cliente, el cual debe proporcionar un prompt, que aceptará únicamente el siguiente comando:

- ***connect usuario@numero_ip:port*** se conecta al servidor que se encuentra en esa *ip* y a ese *puerto* y utilizar ese *usuario*. Luego debe solicitarle el *password* de dicho *usuario*.

Al llegarle esta información al servidor, de no existir el usuario o de haber ingresado un password incorrecto, debe notificarse al cliente, con leyenda “nombre de usuario y/o contraseña incorrecto”. En caso de haber sido positivo, el servidor le envía un mensaje de autorización.

Una vez autenticado, el cliente muestra un nuevo prompt, que identifica al usuario y al servidor que se conectó (*usuario@servidor:~\$*). En este contexto, el usuario puede ingresar comandos que serán enviados al servidor y ejecutados por el *baash*. Toda salida del *baash* debe ser transmitida por el mismo socket al cliente, y se mostrado en pantalla.

Se pide además, que se implemente una función que permita la transferencia de un archivo desde el servidor al cliente. El prototipo de dicha función debe ser el siguiente:

- ***descarga nombre_archivo*** descarga un archivo del servidor al cliente


La transferencia del archivo debe realizarse utilizando conexión no segura (sin conexión) y durante la operación se debe bloquear la interfaz de usuario del programa, de tal forma que no se puedan ingresar nuevos comandos en la aplicación hasta que no haya finalizado la transferencia.

Debe incluirse un mecanismo de control y manejo de errores por parte del servidor con comunicación al cliente.

Todos los procesos deben ser mono-thread y no debe usarse ningún sistema de base de datos.

A fines de llevar a cabo su implementación se debe utilizar como servidor alguna placa de desarrollo y como cliente una computadora cualquiera. Se pone a disposición las placas de desarrollo INTEL Galileo V1, disponibles en en LAC. También el estudiante puede utilizar su propia placa de desarrollo, siempre y cuando soporte MMU y posea un arquitectura compatible con GNU/Linux (como Raspberry Pi).

Tanto la especificación del protocolo de red en la capa de aplicación, así como la elección de la interfaz de usuario del programa cliente, quedan liberadas a criterio del estudiante.



Se exige el uso de Cppcheck y la compilación con el uso de las flags de warning *-Werror*, *-Wall* y *-pedantic*. Se pide utilizar el estilo de escritura de código de GNU [2] o el estilo de escritura del kernel de Linux [3].

Entrega

Se deberá proveer los archivos fuente, así como cualquier otro archivo asociado a la compilación, archivos de proyecto "Makefile" y el código correctamente documentado. También se debe entregar un informe, con el formato adjunto. Se debe asumir que las pruebas de compilación se realizarán en un equipo que cuenta con las herramientas típicas de consola para el desarrollo de programas (Ej: gcc, make), y **NO** se cuenta con herramientas "GUI" para la compilación de los mismos (Ej: eclipse).

Evaluación

El presente trabajo práctico es individual deberá entregarse antes de las **23:50 (UTC -3) del día 22 de Abril de 2018** mediante el LEV. Será corregido y luego deberá coordinar una fecha para la defensa oral del mismo.

Referencias

[1] W. Richard Stevens, Stephen A. Rago, Advanced Programming in the UNIX Environment, 3rd Edition, Addison-Wesley

[2] GNU, "5 Making The Best Use of C",
https://www.gnu.org/prep/standards/html_node/Writing-C.html, [Marzo 2018]

[3] Trovalds, L., Et al., <https://github.com/torvalds/linux/tree/master/Documentation/process>, [Marzo, 2018]

