Sockets de internet en sistemas tipo UNIX

Trabajo práctico Nº 1

Sistemas Operativos II
DC - FCEFyN - UNC

Marzo de 2019

1 Introducción

Los sockets son una abstracción de comunicación entre procesos (IPC) 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.

2 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.

3 Desarrollo

Se pide que diseñe, implemente y testee un software (desarrollado en lenguaje C), que permita realizar la conexión, control y transferencia de un programa servidor con n programas clientes, que simula un satélite geostacionario con su correspondiente estación terrena.

El programa que corren el satélite (cliente) se debe conectar a la estación terrena de forma segura (orientado a conexión), a una IP y puerto fijo. El cliente, debe tomar un número de un puerto libre de su sistema operativo.

El programa que corre en la estación terrena debe proporcionar un prompt, al cual se accede mediante validación de usuario y contraseña que se le solicita al usuario cuando inicia el programa. De ingresar credenciales erróneas, el sistema debe notificar al usuario el usuario y pedirle que las ingrese nuevamente. Al tercer intento incorrecto, debe terminarse el programa.

De ingresar satisfactoriamente al sistema (el prompt debe cambiar) el usuario puede ingresar los siguientes comandos:

- **update firmware.bin** envía un archivo binario al satélite con una actualización del software ("firmware") cliente, subido el archivo, se debe reiniciar el satélite con la nueva actualización.
- **start scanning** inicia un *escaneo* de toda la cara de la Tierra una sección del planeta. Cada escaneo se debe enviar a la estación terrena, que va a estar midiendo el tiempo que le lleva el escaneo de todo el disco (desde que envía el comando, hasta que recibe el último datagrama). Este proceso debe ser lo más óptimo posible. El tamaño de cada escaneo está determinado por el tamaño máximo de un datagrama.
- **obtener telemetría** cada satélite le envía a la estación terrena la siguiente información:
  - Id del satélite
  - Uptime del satélite
  - Versión del software
  - Consumo de memoria y CPU

Se pide que en primer lugar se implemente la solución utilizando sockets UNIX, en una computadora de uso general. Luego implementar todos los programas usando sockets de internet: *Stream* para todos los comandos, excepto *obtener telemetría*, el cual se debe implementar utilizando *Datagram*, la estación terrena en la computadora y los satélites en un sistema embebido.

Debe incluirse un mecanismo de control y manejo de errores en todo el sistema.

Como imagen a escanear se recomienda utilizar [2] y se puede particionar con cualquier script o aplicación. Todos los procesos deben ser mono-thread.

A fines de llevar a cabo su implementación se debe utilizar como servidor una computadora de propósito general y como satélite uno un sistema embebidos. Pueden 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 [3] o el estilo de escritura del kernel de Linux [4].

4 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 incluir gráficas que demuestren la optimización solicitada. 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).

5 Evaluación

El presente trabajo práctico es individual deberá entregarse antes de las 23:50 ART del día 11 de Abril de 2019 mediante el LEV. Será corregido y luego deberá coordinar una fecha para la defensa oral del mismo.

6 Referencias

1. Michael Kerrisk, The Linux Programming Interface, 2010, Adison-Wesley
2. Imagen del satelite Goes16, "https://cdn.star.nesdis.noaa.gov/GOES16/ABI/FD/13/20190811400_GOES16-ABI-FD-13-10848x10848.jpg.zip", Online; accessed 20 Marzo 2019
3. Making The Best Use of C, https://www.gnu.org/prep/standards/html_node/Writing-C.html, Online; accessed 20 Marzo 2019
4. L.Trovalds, Et al., https://github.com/torvalds/linux/tree/master/Documentation/process, Online; accessed 20 Marzo 2019,