Universidad
Nacional
de Córdoba

Cátedra de Sistemas Operativos II

Trabajo Práctico Nº I

NAVARRO, Matias Alejandro

11 de abril del 2019

Índice

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 envı´a o recibe información, al igual que como se lee o escribe un archivo. Son una herramienta muy importante, uno de los pilares de la comunicación entre procesos, y sumamente utilizada en la mayorı´a de las aplicaciones de red.

Propósito

El propósito de este Trabajo Práctico es aprender a utilizar la estructura de sockets de Linux, sus características y limitaciones, el uso de sus diferentes tipos (orientados o no a la conexión), y su posterior implementación y testing.
Los conceptos deben aplicarse a un caso particular referido a una estación terrena (servidor) que debe interactuar con un satélite (cliente), y entre los mismos enviar o recibir distintos datos, como la información del satélite, archivos de actualización de firmware o la imagen de la tierra vista desde el satélite.

Ámbito del Sistema

El sistema está conformado por dos partes: la aplicación servidor, que se ejecuta desde la computadora del usuario, y la aplicación cliente, que se ejecuta en una placa Raspberry Pi 3. Ambos comunicándose por medio de sockets enviando y recibiendo los distintos datos.

Definiciones, Acrónimos y Abreviaturas

- TCP: Transmission Control Protocol (Protocolo de Control de Transmission).
- UDP: User Datagram Protocol (Protocolo de Datagramas de Usuarios).
- Sockets: "canal de comunicación" entre dos programas que corren sobre ordenadores distintos o incluso en el mismo ordenador.
- INET: sockets de internet.

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/13/20190811400GOES16-ABI-FD-13-10848x10848.jpg.zip”
3. Making The Best Use of C, https://www.gnu.org/prep/standards/html node/Writing-C.html
4. L.Trovalds, Et al., https://github.com/torvalds/linux/tree/master/Documentation/process

Descripción General del Documento

El propósito de este documento es proporcionar a quien lo lea un entendimiento básico de los requisitos y alcance de este proyecto así como los pasos que se tomaron para el diseño e implementación al proyecto propuesto.

Descripción General

Perspectiva del Producto

El producto está orientado a la comunicación de satélites con las estaciones terrestre para que estas puedan recibir las distintas imágenes satelitales. Y en caso de ser necesario actualizar el firmware del satélite desde la tierra.

Funciones del Producto

El producto provee las siguientes funciones:

- Conectividad usuario servidor mediante IP y puerto.
- Autentificación para brindar seguridad en los datos.
- Control de errores de autentificación (3 permitidos).
- Actualización remota de firmware de satélite.
- Envió de las imágenes satelitales a la estación.
- Obtención de los datos del satélite (telemetría).

Características de los Usuarios

El usuario son los astrólogos situados en las distintas estaciones/observatorios con una computadora con alguna distribución de Linux (las pruebas se hicieron en Linux Mint/Ubuntu) y posea un conocimiento básico para manejarse con la consola ya que no se cuenta con una interfaz gráfica.

Restricciones

La licencia del producto pertenecera al gobierno. No se podrá hacer uso del código realizado para su venta a instituciones nacionales o extranjeras.

Suposiciones y Dependencias

- El servidor se inicializa en un puerto fijo (6020).
- El proyecto se ejecutará sobre Linux.
- Existe conectividad entre el servidor y el cliente, para poder realizar la comunicación.
- No hay ninguna dependencia específica, el proyecto debe funcionar en una distribución Linux estándar utilizando *make* para compilar el código fuente.
- Tanto el servidor como el cliente tendrán acceso a internet y tendrán una dirección Ipv4.
- El servidor dispondrá de memoria suficiente para almacenar las distintas imágenes satelitales.

Requisitos Específicos

Interfaces Externas

- Dispositivo del cliente que debe tener instalado un sistema operativo con kernel linux.
- Dispositivo del servidor que debe tener instalado un sistema operativo con kernel linux.

Funciones

Una vez que el usuario se ha autentificado en el servidor, el mismo acepta los siguientes comandos:

- **update firmware**: envía un archivo binario al satélite con una actualización de software (“firmware”), una vez actualizado, se debe reiniciar el satélite con la nueva actualización.
- **get telemetría**: el satélite le envía a la estación terrena la siguiente información:
  - ID: Nombre del satélite.
  - Uptime: hora y fecha de la última actualización.
  - Versión: versión actual de firmware.
  - CPU: consumo de memoria y CPU.
- **start scanning**: inicia un *escaneo* de toda la cara de la Tierra. Cada escaneo se envía a la estación terrena.
- **help**: muestra un mensaje de ayuda con los comandos disponibles.

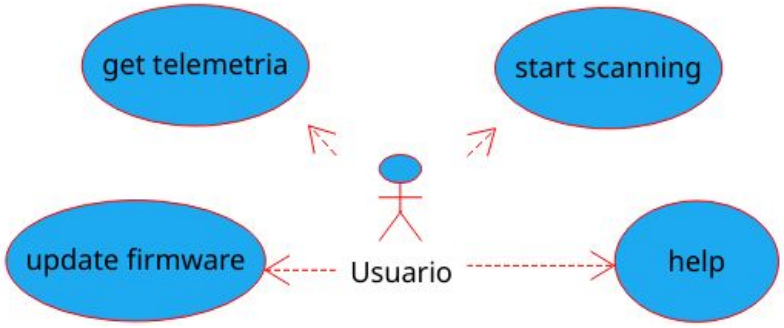

Imagen 1 - Diagrama de Casos de Uso

Requisitos de Rendimiento

El proceso de enviar la imagen del satélite a la estación terrena debe ser lo más óptima y en el menor tiempo posible.

Restricciones de Diseño

- En el servidor no se podrán conectar más de un usuario por vez.
- El servidor solo se comunica con un satélite.
- Para la compilación es recomendable utilizar flags como -Werror, -Wall y -pedantic.

Atributos del Sistema

El servidor debe correr en una computadora con distribución Linux, mientras que el cliente (satélite) debe correr sobre una Raspberry Pi.

Diseño de solución

Para el diseño de la solución al problema planteado se lo abordó dividiéndolo en partes más pequeñas, de esta forma se diseñó la comunicación con sockets TCP por un lado, la comunicación con sockets UDP y las funciones por otro.

Se desarrollaron dos aplicaciones, una que es ejecutada en el cliente y otra en el servidor.

- *Cliente*: se conecta al servidor y queda esperando los comandos que son introducidos desde el servidor para realizar las distintas funciones.
- *Servidor*: crea los sockets para la conexión, autentifica los usuarios que luego podrán pedir un dato específico al cliente mediante el ingreso de comandos disponibles.

Implementación y Resultados

Para la implementación y desarrollo se utilizó el software *Visual Studio Code* y los códigos sockets aportados por la cátedra. A su vez también se crearon funciones y librerías las cuales se utilizaron para dicho proyecto.
Primero se procedió a trabajar e implementar el trabajo sobre sockets UNIX tanto el cliente como el servidor. Una vez que el trabajo estuvo implementado en dichos sockets, se comprobó la funcionalidad de los mismos para comprobar que todo funcionara como debería.
Luego, una vez terminado el trabajo de UNIX se pasó a implementar el desarrollo sobre sockets INET y definiendo el puerto fijo de comunicación 6020.

Para el análisis del código (implementado en C) se utilizó la herramienta *cppcheck* y la compilación del proyecto se realizó con *gcc* con flags *-Werror*, *-Wall* y *-pedantic*.

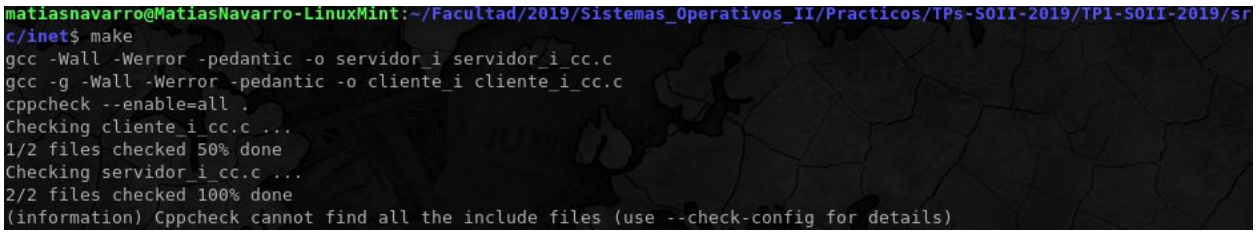

Imagen 2 - Compilación y Análisis del Código

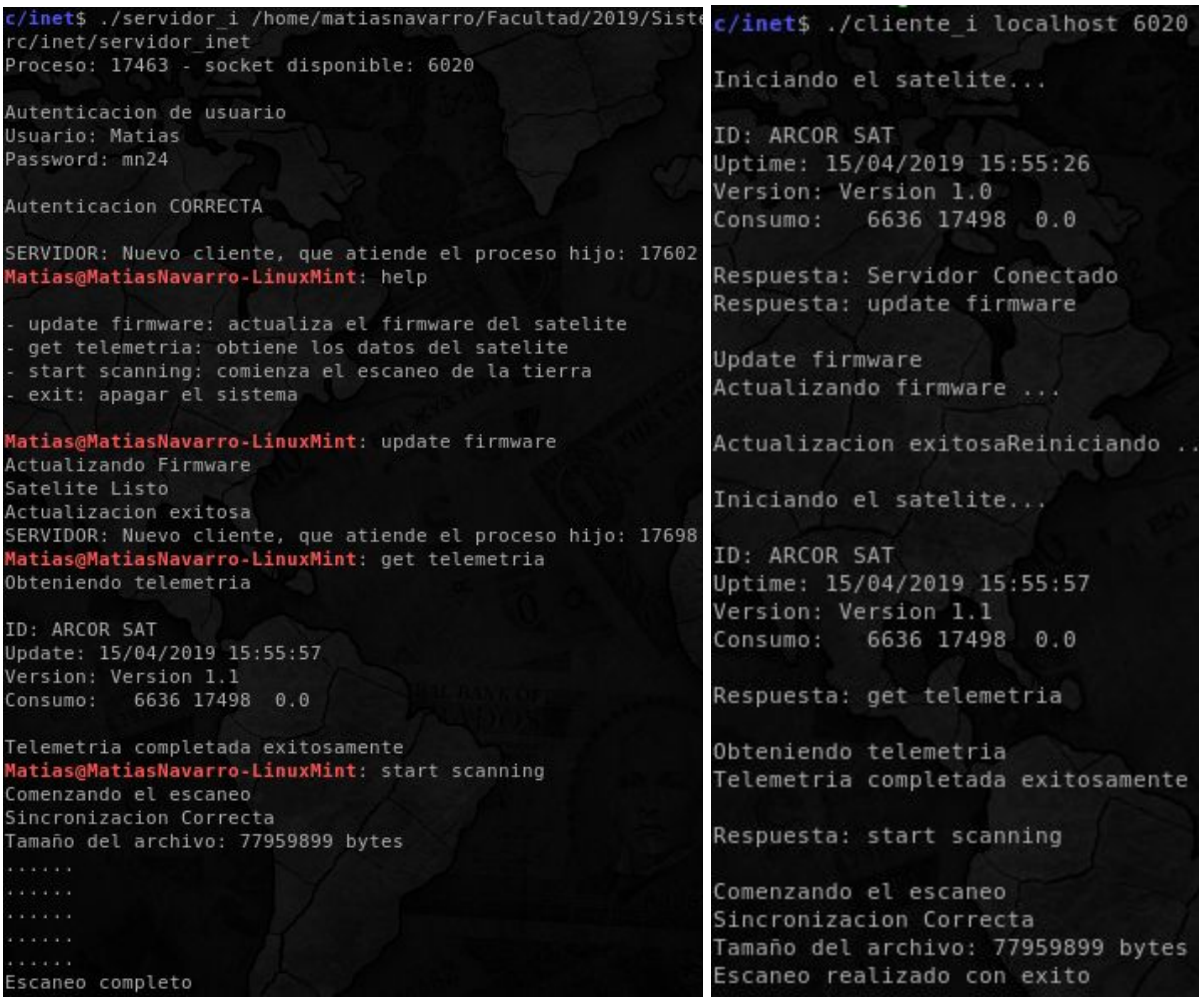

Imagen 3 - Programa en funcionamiento.
La imagen izquierda corresponde al servidor, mientras que la imagen izquierda pertenece al cliente.

Conclusiones

Por medio de la resolución del trabajo práctico se logró llevar a cabo una conexión de dos terminales distintas por medio de un socket. Logrando así llevar a cabo un proyecto de la vida real.

Además, se profundizaron conocimiento de programación en c, uso de punteros, acceso a archivos entre otros.

Apéndices

A la hora de realizar este trabajo se utilizó contenido de las siguientes URL:

- http://www.holamundo.es/lenguaje/c/articulos/fecha-hora-c.html
- https://www.geeksforgeeks.org/socket-programming-cc/
- https://www.binarytides.com/programming-udp-sockets-c-linux/
- http://www.chuidiang.org/clinux/sockets/sockets_simp.php