Tarea 1: Desarrollo de un 'Daemon' para Linux

Daniel Cob-Beirute
Escuela de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica
Cartago, Costa Rica
danielcob@estudiantec.cr

Adriel Sebastián Chaves Salazar
Escuela de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica
Cartago, Costa Rica
adriel.chaves@estudiantec.cr

I. Desarrollo de las preguntas de aprendizaje continuo:

A. Pregunta#1 : ¿Cuáles son las necesidades actuales de aprendizaje para enfrentar el proyecto?

Fue necesario reforzar conocimientos en programación en C, particularmente en el manejo de sockets y comunicación cliente-servidor, ya que la arquitectura se basa en este paradigma. Además, se requirió aprender a trabajar con hilos para mantener un proceso en escucha permanente y con systemd para gestionar el daemon que levanta el servicio de forma automática. También se revisaron fundamentos de procesamiento de imágenes como la ecualización de histograma y la clasificación de colores predominantes.

B. Pregunta#2 : ¿Cuáles son las tecnologías que se pueden utilizar para el desarrollo?

Las tecnologías empleadas fueron el lenguaje C con el soporte de la biblioteca POSIX para el manejo de procesos e hilos, así como sockets TCP para la comunicación. Para la ejecución del servidor como servicio se utilizó systemd. Se implementó un archivo de configuración que define rutas de salida, puerto de comunicación y logs, junto con un esquema de directorios para la clasificación de imágenes. El cliente se ejecutó en una máquina virtual, siguiendo la recomendación del curso.

C. Pregunta#3 : ¿Cuáles acciones se implementó para el desarrollo del proyecto (organización de tiempo, búsqueda de información, repaso de contenidos, entre otros)?

Se definieron tareas en etapas: primero el desarrollo del servidor, después la lógica de procesamiento de imágenes y finalmente el cliente. Se utilizó un repositorio en GitHub para llevar control de versiones y dividir el trabajo entre los miembros del equipo. Se recurrió a documentación oficial de Linux, ejemplos de sockets en C y tutoriales de systemd. Además, se dedicó tiempo al repaso de conceptos de redes y sistemas operativos previamente estudiados.

D. Pregunta#4 : Evalúe de forma crítica la eficiencia de las acciones implementadas en el contexto tecnológico

La planificación mediante GitHub y la división de responsabilidades permitieron mantener orden en el proyecto. Sin embargo, la integración del procesamiento de imágenes tomó más tiempo de lo esperado, debido a la necesidad de adaptar algoritmos al formato utilizado. El uso de systemd fue acertado ya que permitió automatizar pruebas y facilitar la ejecución del servidor, aunque requirió un aprendizaje adicional para configurar correctamente el archivo de servicio.

II. Desarrollo de las preguntas de trabajo individual y en equipo:

A. Pregunta#1 : Indicar las estrategias para el trabajo individual y en equipo de forma equitativa e inclusiva en las etapas del proyecto (planificación, ejecución y evaluación).

Cada integrante asumió un rol diferenciado: uno se centró en el desarrollo del servidor y el otro en el cliente. Esta separación favoreció la equidad en la carga de trabajo y aseguró que ambos pudieran aportar según sus fortalezas técnicas.

B. Pregunta#2 : Indicar la planificación del trabajo mediante la identificación de roles, metas y reglas.

Se planificó con metas semanales: primero establecer comunicación cliente-servidor, luego el procesamiento de imágenes y finalmente la integración con systemd. Como regla, todo avance debía subirse al repositorio GitHub con mensajes de commit claros.

C. Pregunta#3 : Indicar cuáles acciones promueven la colaboración entre los miembros del equipo durante el desarrollo del proyecto.

El uso de GitHub fue fundamental para la colaboración. Además, se hicieron revisiones cruzadas del código: cada integrante probaba la parte implementada por el otro. Esto permitió detectar errores temprano y garantizar compatibilidad entre los módulos.

D. Pregunta#4 : Indicar cómo se ejecutan las estrategias planificadas para el logro de los objetivos.

Las estrategias se ejecutaron de forma secuencial: comunicación básica, luego funcionalidades del servidor, pruebas con el cliente y finalmente empaquetado del servicio. Las revisiones periódicas aseguraron que se cumplieran los objetivos de cada fase.

E. Pregunta#5 : Indicar la evaluación para el desempeño del trabajo individual y en equipo.

El desempeño individual fue evaluado por la calidad de cada módulo desarrollado (servidor y cliente). A nivel de equipo, se valoró la integración final del sistema, la cual se logró con éxito tras resolver detalles de compatibilidad en la comunicación.

F. Pregunta#6 : Indicar la evaluación para las estrategias utilizadas de equidad e inclusión.

La distribución de responsabilidades fue equitativa y se mantuvo una comunicación constante. Se procuró que ambos miembros pudieran participar en la documentación y pruebas, no solo en la programación.

G. Pregunta#7 : Indicar la evaluación para las acciones de colaboración entre los miembros del equipo.

Las acciones de colaboración fueron efectivas: el repositorio compartido y las pruebas cruzadas aseguraron consistencia. Aunque hubo retrasos en la integración del procesamiento de imágenes, el trabajo en conjunto permitió superarlos a tiempo.

III. Breve explicación del uso de systemd

Systemd es un sistema de inicialización y gestor de servicios ampliamente utilizado en las distribuciones modernas de Linux. Su objetivo principal es administrar los procesos que se ejecutan en segundo plano (demonios) y controlar el arranque del sistema de manera más eficiente que los sistemas previos como SysVinit [1], [2].

Una de sus características más relevantes es la capacidad de definir servicios mediante archivos de configuración con extensión `.service`, los cuales permiten especificar dependencias, comandos de inicio, reinicio automático y rutas de logs. Esto facilita el despliegue de aplicaciones que requieren ejecutarse en segundo plano, como servidores web o demonios de procesamiento de imágenes, tal como el proyecto desarrollado en esta tarea.

Además, systemd incluye el comando `systemctl`, que permite a los administradores gestionar los servicios con acciones como `start`, `stop`, `restart` y `status`. Esto provee un mecanismo uniforme y confiable para el control del ciclo de vida de los demonios [3].

En el contexto de este proyecto, el uso de systemd permitió que el servidor `ImageServer` se iniciara automáticamente al encender la máquina, asegurando que las dependencias de red estuvieran listas antes de que el servicio comenzara a aceptar conexiones. Esta integración no solo aumentó la estabilidad del sistema, sino que también simplificó las pruebas y validaciones realizadas en diferentes entornos virtualizados.

IV. Interfaz gráfica del cliente

Para facilitar el uso del cliente se desarrollo una interfaz gráfica usando gtk. Esta permite al usuario cargar de forma simple las imágenes, seleccionando que tipo de procesamiento se quiere realizar (ecualización por histograma, clasificación por color o ambas). Además la interfaz permite ajustar el archivo de configuración de forma simple sin tener que entrar a la carpeta del proyecto. El resultado se puede observar en la figura 1.

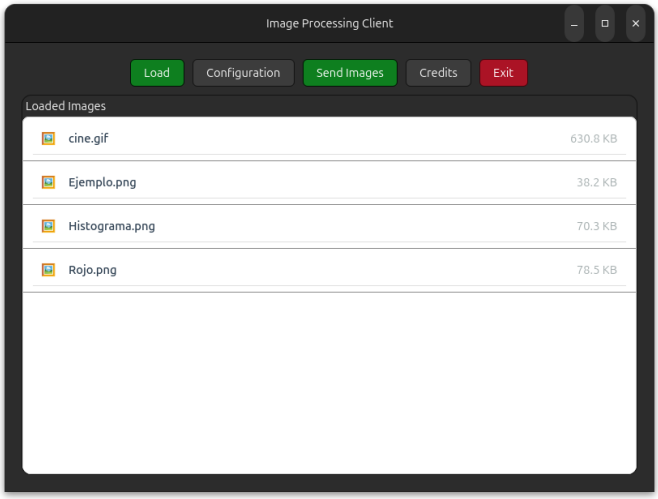

Fig. 1. Interfaz gráfica del cliente desarrollada en el proyecto.

References

[1] "systemd - system and service manager," https://freedesktop.org/wiki/Software/systemd/, accedido: Sept. 2025.

[2] L. Poettering, "systemd for administrators," in *Linux Plumbers Conference*, 2010.

[3] "systemd documentation," https://www.freedesktop.org/software/systemd/man/systemd.service.html, accedido: Sept. 2025.