**Instituto Tecnológico de Costa Rica**
Área Académica de Ingeniería en Computadores
CE-4303 Principios de Sistemas Operativos


# Tarea I
## Filtrador y clasificador de imágenes como servicio

Elaborado por:

Bryan Masís Mora
Cristofer Fernández Fernández
Alejandra Castrillo Muñoz 2015155759

Profesor:
Jason Leitón Jiménez

Cartago
II Semestre 2019

## Introducción

Los ‘demonios’ o ‘daemons’, por su nombre en inglés corresponden a procesos de una computadora que se ejecutan en segundo plano, esto tiene varias implicaciones. Como bien es conocido, la mayor parte de las funciones básicas de un sistema operativo como la administración de recursos de memoria, red, dispositivos de entrada y salida entre otros es gestionado por servicios en segundo plano o daemons (Tanenbaum, A. S., 2003). La función de ecualizador de una imagen trabaja a nivel de píxel sobre esta y su resultado es una imagen derivada de la original con detalles mejorados dado que resalta el contraste .

En el escrito se diseña e implementa 3 componentes básicos, el servidor REST basado en el lenguaje de programación C que es el encargado de realizar la función de ecualización, priorización de trabajo y clasificación de imágenes. La función de priorización se basa en ejecutar primero el archivo más pequeño, entonces si ya se está ejecutando un proceso y se recibe una imagen más pequeña, se mata el proceso y se prioriza el nuevo. La función de clasificación divide las imágenes de entrada según la cantidad de colores básicos que estas tengan basado en RGB.

Para la implementación del servicio se utilizó *systemd*, este permite la creación y administración de procesos. Este soporta comandos de SysV y ofrece más herramientas que otros sistemas.

## Ambiente de desarrollo

La totalidad del proyecto fue desarrollada en Linux, Ubuntu en su versión 18.04 de forma más concreta.
La implementación de los algoritmos del server están escritas en el lenguaje C, mientras que el cliente se implementó con la ayuda de Ionic y angularJS.

## Atributos

- Herramientas de Ingeniería (HI): Avanzado.
- Aprendizaje Continuo(AC): Avanzado.
- Ingeniería y sociedad (IS): Intermedio.

## Diseño

## Instrucciones de uso

Correr clientes:

- Ionic:
    - Ionic seve
- Angular JS 7+:
    - Ng serve

Correr el servidor C

```
$ Gcc -o server server.c

$./server {puerto}
```

Instalación de bibliotecas

```
$ sudo apt install python-pip

$ pip install Wand
```

Iniciar

```
$ chmod +x run.sh

$ ./run.sh
```

Instalar del demonio:

```
$ ./scripts/install-daemon.sh
```

Desinstalar demonio:

```
$ ./scripts/uninstall-daemon.sh
```

Controlar ejecución del demonio

```
$ systemctl start ImageServer.service
$ systemctl status ImageServer.service
$ systemctl stop ImageServer.service
$ systemctl reload ImageServer.service
```

## Coevaluación

## Bitácoras

| Persona | Actividad | Descripción |
|---|---|---|
| Bryan Masís | Instalación de Ubuntu | 22/8/19<br>Mi laptop no posee ubuntu instalado y debido a la naturaleza de nuestro proyecto, es necesario. Por lo que procedí a instalar el sistema operativo en booteo doble. Encontré múltiples problemas en este proceso, el instalador crashea siempre al escoger el idioma y tuve que investigar cómo solucionar este problema. Resolver este error tomó |

| | | |
|---|---|---|
| | | aproximadamente 5hrs. |
| Bryan Masís | Diseño del socket server | 24/8/19<br>Investigué cómo realizar un socket http en C, que me permitiera escalarlo para que cumpliera con los requisitos del proyecto.<br>Duré aproximadamente 4hrs en esto. |
| Bryan Masís | Revisión al server | 28/8/19<br>En la implementación actual existen problemas con el standard HTTP a la hora de realizar los request, por lo que se debe cambiar la estructura del mismo.<br>Esto tomó aproximadamente 3hrs. |
| Bryan Masís | Implementación del método para recepción de imágenes | 30/8/19<br>Una vez definidas las rutas de POST y GET, implementé un método para recibir y enviar las imágenes.<br>El método de GET funciona perfecto y se encarga de enviar la imagen que se le pide como argument.<br>El método POST tiene problemas actualmente, recibe la imagen parcialmente, además dentro de los datos de la misma, mete el header. |
| Bryan Masís | Correcciones del método POST | 2/9/19<br>Logré que el método acepte la imagen |

| | | |
|---|---|---|
| | | completa, sin embargo, tengo el mismo problema de que recibe el header de la imagen, no se cómo parsearla sin perder datos en el proces. (bytes mezclados puesto que el delimitador del chunck de bytes por iteración del método puede o no calzar con un byte de datos) |
| Bryan Masís | Correciones del método POST | 3/9/19<br>Todavía no encuentro solución al problema, decidí investigar sobre formas de aplicar el ecualizador de histograma a una imagen dada mientras tanto, encontré un método que realiza el proceso pero necesita las dimensiones de la imagen, así como que esta se encuentre en blanco y negro. Por el momento decidí implementarlo para tener algo en ese apartado. Encontré una librería llamada MagicWand para C la cual se encarga de procesamiento de imágenes, por lo que pronto espero implementarla. |
| Bryan Masís | Trabajo en el método POST | 4/6/19<br>Nuevamente busco formas de parsear este header fuera de los bytes de la imagen, encontré un método de escritura de datos pero tenía el mismo |

| | | |
|---|---|---|
| | | problema del primero, no se guardan los datos sin importar el tamaño del buffer.<br>Sigo buscando soluciones. |
| Bryan Masís | Arreglos del socket | 6/9/19<br>Utilizando magicwand se puede hacer la ecualización de forma inmediata, sólo dandole el nombre del archivo, por lo que decidí implementarla en el server.<br>Nuevamente seguí tratando de solucionar el problema con el header pero no hubo suerte, por lo que decidí aportar a la documentación. |
| Crisptofer | Crear Cliente web | 03/09/2019<br>Implementación del cliente web en Ionic pero fue fallido. |
| Crisptofer | Utilizar Angular para cliente web | 04/09/2019<br>Se cambió de plataforma a angular para crear el cliente web |
| Crisptofer | Migrar angular a ionic | 08/09/2019<br>Migrar el codigo de angular a ionic para generar el cliente movil. |
| Crisptofer | Arreglar problemas de conexión | 09/08/2019<br>Se repararon problemas de conexión entre el server y los clientes |
| | | |
| | | |
| Alejandra Castrillo | Información sobre | 30/08/2019 |

| | | |
|---|---|---|
| | plataformas | Investigación sobre las diferentes opciones para administrar procesos. |
| Alejandra Castrillo | Investigación de creación de demonios | 31/08/2019<br>Investigación sobre el proceso de creción de demonios en linux. |
| Alejandra Castrillo | Creación de procesos de prueba | 31/08/2019<br>Creación de procesos con systemctl para entender el funcionamiento de sus scripts y comandos. |
| Alejandra Castrillo | Creación de demonio con c. | 1/08/2019<br>Creación de demonio con systemctl y c con el repositorio del ejemplo mostrado en las referencias. |
| Alejandra Castrillo | Investigación sobre Make | 1/08/2019<br>Aprender a utilizar Make para simplificar ejecución del programa. |
| Alejandra Castrillo | Creación de demonio con Make | 1/08/2019<br>Creación de demonio con archivo make para ejecución. |
| Alejandra Castrillo | Unión de server con el demonio | 2/08/2019<br>Unión del demonio con un servidor de prueba. |
| Alejandra Castrillo | Arreglo de errores | 4/08/2019<br>Arreglo de errores de implementación con el servidor actual y el demonio |
| Alejandra Castrillo | Unión del servidor final con el demonio | 6/08/2019<br>Implementación del servidor final con el demonio hecho. |

## Conclusiones

- Se determina que los procesos en segundo plano o Daemons son de suma importancia en un sistema dado que proveen funciones necesarias para el este sin tener que iniciar explícitamente una aplicación con GUI.
- Ionic es un sistema multiplataforma muy poderoso para generar aplicaciones web y móvil, dado que permite realizar el trabajo una sola vez y de esta forma no tener que diseñar 2 veces la misma solución para diferentes plataformas.
- MagickWand es una biblioteca muy poderosa para el manejo de imágenes en varios lenguajes como C, C++, PHP. Su instalación es algo engorrosa pero vale la pena el esfuerzo.
- C como lenguaje tiene muchas limitantes, y estas salen a flote cuando se trata de lidiar con protocolos de comunicación por internet.

## Sugerencias y recomendaciones

- Se sugiere profundizar más en el tema de daemons dado que es una herramienta poderosa a la hora de diseñar sistemas cliente servidor.
- Si se requiere manipular imágenes, ImageMagick es una gran opción.

## Referencias

Tanenbaum, A. S. (2003). Sistemas operativos modernos. Pearson Educación.

Hnidek, J. Example of Linux Daemon. Tomado de: https://github.com/jirihnidek/daemon.git.

Pid Ein (2011). Why systemd?. Tomado de: http://0pointer.de/blog/projects/why.html