

Sistemas Operativos 1/2018

Laboratorio 1

Profesores:

Cristóbal Acosta (cristobal.acosta@usach.cl)
Miguel Cárcamo (miguel.carcamo@usach.cl)
Fernando Rannou (fernando.rannou@usach.cl)

Ayudantes:

Esteban Alarcón (esteban.alarcon.v@usach.cl)
Marcela Rivera (marcela.rivera.c@usach.cl)

I. Objetivos Generales

Este laboratorio tiene como objetivo refrescar los conocimientos de programación imperativa, mediante la construcción de una aplicación simple de procesamiento de imágenes. La aplicación debe ser escrita en lenguaje de programación C sobre sistema operativo Linux. Este laboratorio servirá de base a los siguientes laboratorios, los cuales irán paulatinamente incluyendo el uso de servicios del sistema operativo para realizar la misma función que la aplicación original. Es por eso que es de vital importancia que el diseño y modularización de esta aplicación quede bien definida a partir del lab1, para que luego sólo nos preocupemos de incluir las nuevas funcionalidades asociadas a procesos, hebras y quizá memoria compartida.

II. Objetivos Específicos

1. Conocer y usar las funcionalidades de `getopt()` como método de recepción de parámetros de entradas.
2. Construir funciones de lectura y escritura de archivos binarios usando `open()`, `read()`, y `write()`.
3. Construir funciones de procesamiento de imágenes
4. Practicar técnicas de documentación de programas
5. Conocer y practicar uso de `makefile` para compilación de programas.

III. Desarrollo

La aplicación consiste en un *pipeline* de procesamiento de imágenes astronómicas. Cada imagen pasará por tres etapas de procesamiento tal que al final del *pipeline* se clasifique la imagen como satisfaciendo o no alguna condición a definir. El programa procesará varias imágenes, una a la vez.

Las etapas del *pipeline* son:

1. Lectura de imagen RGB
2. Conversión a imagen en escala de grises
3. Binarización de imagen
4. Análisis de propiedad
5. Escritura de resultados

III.A. Lectura de imágenes

Las imágenes estarán almacenadas en binario con formato bmp. Habrá n imágenes y sus nombres tendrán el mismo prefijo seguido de un número correlativo. Es decir, los nombres serán `imagen_1`, `imagen_2`, ... , `imagen_n`.

Para este laboratorio se leerá una nueva imagen cuando la anterior haya finalizado el *pipeline* completo. Es decir, en el *pipeline* sólo habrá una imagen a la vez.

III.B. Conversión de RGB a escala de grises

En una imagen RGB, cada pixel de la imagen está representado por tres números, que representan el componente de color rojo (Red), el de color verde (Green) y el de color azul (Blue). Para convertir una imagen a escala de grises, se utiliza la siguiente ecuación de luminiscencia:

$$Y = R * 0.3 + G * 0.59 + B * 0.11 \quad (1)$$

donde R , G , y B son los componentes rojos, verdes y azul, respectivamente.

III.C. Binarización de una imagen

Para binarizar la imagen basta con definir un umbral, el cual define cuáles píxeles deben ser transformados a blanco y cuáles a negro, de la siguiente manera. Para cada pixel de la imagen, hacer

```
si el pixel > UMBRAL
    pixel = 1
sino
    pixel = 0
```

Los valores 0 y 1 no son representados por un bit sino son valores enteros (`int`) Al aplicar el algoritmo anterior, obtendremos la imagen binarizada.

III.D. Clasificación

Se debe crear una función que concluya si la imagen es *nearly black* (casi negra). Esta característica es típica de muchas imágenes astronómicas donde una pequeña fuente puntual de luz está rodeada de vacío u oscuridad. Entonces, si el porcentaje de píxeles negros es mayor o igual a un cierto umbral la imagen es clasificada como *nearly black*.

El programa debe imprimir por pantalla la clasificación final de cada imagen y escribir en disco la imagen binarizada resultante.

IV. Otros conceptos

Como ayuda, la manera de leer una imagen es la siguiente:

- Abrir el archivo con el uso de `open()`. Recuerde que este archivo debe contener la matriz (imagen).
- Leer la imagen con `read()`.

Con los pasos previamente descritos, se puede leer la imagen y manipularla de tal forma que se pueda crear la matriz. Recuerde que la imagen tiene una estructura para poder obtener el valor de los píxeles, por lo que se pide investigar al respecto.

Para el caso de escribir la imagen resultante:

- Se debe escribir con `write()`.
- Finalmente se debe cerrar el archivo con `close()`.

Por último, el programa debe recibir la cantidad de imágenes a leer, el valor del umbral de binarización, el valor del umbral de negrura y una bandera que indica si se desea mostrar o no la conclusión final hecha por la tercera función. Por lo tanto, el formato `getopt` es:

- -c: cantidad de imágenes
- -u: UMBRAL para binarizar la imagen.
- -n: UMBRAL para clasificación
- -b: bandera que indica si se deben mostrar los resultados por pantalla, es decir, la conclusión obtenida al leer la imagen binarizada.

Por ejemplo, la salida por pantalla al analizar 3 imágenes sería lo siguiente:

```
$ ./pipeline -c 3 -u 50 -n 10000 -b
|   image   |   nearly black   |
|-----|-----|
| imagen_1  |         yes      |
| imagen_2  |         no       |
| imagen_3  |         no       |
```

Donde sólo la imagen_1 fue clasificada como *nearly black*.

V. Entregables

Debe entregarse un archivo comprimido que contenga al menos los siguientes archivos:

1. *Makefile*: archivo make para que compile los programas.
2. uno o mas archivos con el proyecto (*.c, *.h).
3. Clara documentación, es decir, entregar códigos debidamente comentados, en donde a lo menos se incluya la descripción de cada función, sus parámetros de entrada y salida.

El archivo comprimido debe llamarse: RUTESTUDIANTE1.RUTESTUDIANTE2.zip

Ejemplo: 19689333k_186593220.zip

NOTA: los laboratorios son en parejas, las cuales deben ser de la misma sección. De lo contrario no se revisarán laboratorios.

VI. Fecha de entrega

Domingo 15 de Abril.